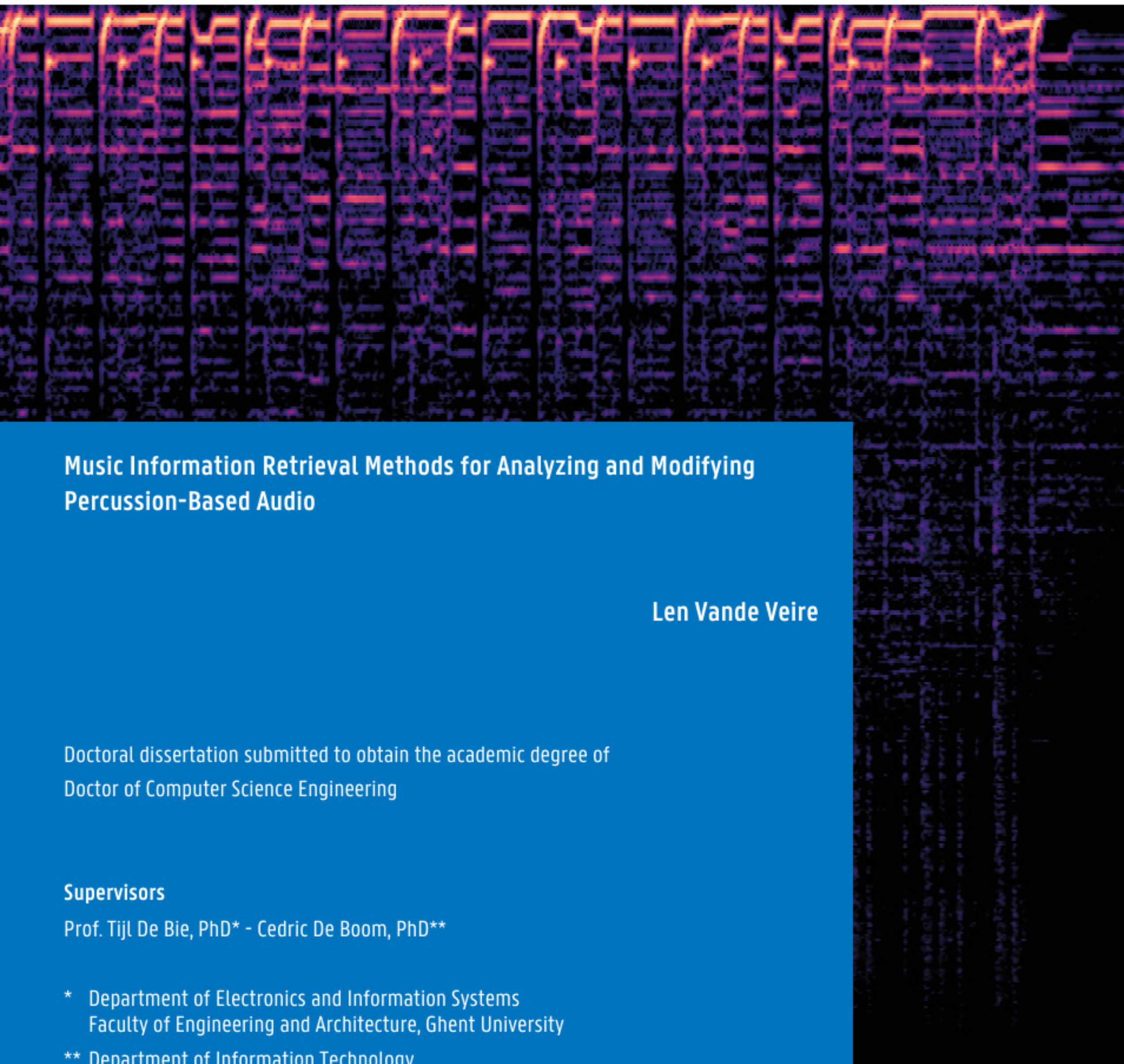
FACULTY OF ENGINEERING
AND ARCHITECTURE



Music Information Retrieval Methods for Analyzing and Modifying Percussion-Based Audio

Len Vande Veire

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Tijl De Bie, PhD* - Cedric De Boom, PhD**

* Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

** Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

August 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

Music Information Retrieval Methods for Analyzing and Modifying Percussion-Based Audio

Len Vande Veire

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Tijl De Bie, PhD* - Cedric De Boom, PhD**

* Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

** Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

August 2021

GHENT
UNIVERSITY

ISBN 978-94-6355-512-8
NUR 980, 984
Wettelijk depot: D/2021/10.500/60

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Thomas Demeester, PhD, Ghent University
Prof. Jason Hockman, PhD, Birmingham City University, United Kingdom
Johan Pauwels, PhD, Imperial College London, United Kingdom
Edith Van Dyck, PhD, Ghent University

Supervisors

Prof. Tijl De Bie, PhD, Ghent University
Cedric De Boom, PhD, Ghent University

Word of Thanks

This thesis is an academic dissertation, and it therefore describes the results of my four years of PhD research. What is not evident from that discussion, however, is the journey behind these results – the ups and downs, the strokes of luck and the setbacks, the ideas that didn't work or that didn't make the light of day (of which there were many), the moments of insight and excitement and those of frustration and sometimes even despair. This dissertation would not be complete without an acknowledgement of this journey. This project has been truly fascinating, and at the same time it has challenged me in ways that I had not anticipated when I started it four years ago.

And challenging though it may have been, I have – evidently – made it to the end. This is an achievement that would not have been possible without the support of many people, and I would like to take this opportunity to say thanks.

First and foremost, I would like to thank my promoters Tijl and Cedric for being my guides and mentors during the past two-and-a-half years. I am immensely grateful that I could always rely on you for your guidance, counsel and patience. Cedric, it has been a lot of fun to work together with someone who shares my passion for music and DJing, and I have very much enjoyed our many interesting conversations and thought-provoking brainstorms. Tijl, I consider myself very lucky to have had such a skilled, kind and supportive mentor for both my Master's thesis and my PhD research. Thank you for that. I would also like to explicitly thank you for

convincing me a little over two years ago to submit my work on the drum'n'bass style transfer system to the ML4MD workshop at ICML 2019; I consider that moment to be a crucial turning point in this journey, and I am very grateful for how that and all subsequent events unfolded.

I did not start my PhD under the supervision of Tijl and Cedric, though. Joni, Francis, thank you as well for guiding me during the first one-and-a-half years of my PhD. I am very grateful that you gave me the freedom to explore new avenues of research, and that you supported me throughout this process, even if this meant that eventually I would go my own way after a while.

Research rarely happens in a vacuum, and mine certainly did not either. I would therefore like to thank all the other researchers that I have had the privilege of working together with in one way or another. Firstly, I would like to thank the members of my doctoral examination committee for their sharp and thorough comments and for the interesting discussions that we have had. I am convinced that the new perspectives that I gained thanks to your feedback have led to substantial improvements in this dissertation. I would also like to thank my fellow researchers – in and outside of Ghent University – for the many interesting discussions and hackathons; for your feedback on and thought-provoking questions about my research; and for the enjoyable company at work and at conferences. Thanks also to my closest colleagues at Ghent University, both from my current and from my previous research group. In some sense I was often the "odd one out" – being the only researcher with such a strong focus on MIR – and I am very happy that I could still enjoy your support and your company.

My passion for music of course did not sprout from nothingness either. I therefore want to express my boundless appreciation for a group of people who have not actively or knowingly helped me, but without whom this project would not have been the same: the

artists, DJs, music educators and in general all the people in the music community that have in some sense accompanied me throughout these years. I have listened to countless hours (or days, or weeks) of DJ sets and music in the past four years: while programming and debugging, while writing or rewriting papers, while brainstorming new research ideas, or when taking a break from all that. Your music has been an inexhaustible source of energy and inspiration and a catalyst that helped me to channel my energy and focus into my work. Thank you for doing what you do – your music is making the world a more beautiful place.

And last but definitely not least, a big hug goes to my friends and my family. With you I could share my excitement when things went well and my worries when things got rough. You were always there with advice and support when I needed it; and all the fun moments we had together have kept me happy and sane. Special thanks go to my mom: because you are always and unconditionally there for me. Words cannot describe how grateful I am for that. And to Sofie: ever since I met you two summers ago, you have brought laughter and joy into my life. I am very happy that you are here by my side, and I'm looking forward to the many happy moments that we will share in the future.

* * *

At the end of this journey, I must come to the conclusion that my life has become permeated by music. This journey has strengthened a passion that I am now sure is here to stay. A new world has opened for me, one that I am extremely excited to be a part of and one that I cannot wait to continue to explore.

Thank you all for making this possible. I am forever grateful.

Len
August 2021

Len Vande Veire performed his research as a member of the Artificial Intelligence and Data Analytics (AIDA) Research Group, IDLab, Department of Electronics and Information Systems at Ghent University.

For this research, Len Vande Veire was supported by a PhD fellowship of the Fonds Wetenschappelijk Onderzoek—Vlaanderen (file No. 41856).

Summary

We live in an age of digitization, and this enables the realization of many interesting applications to process and interact with music. One example of such an application is the fast and easy retrieval of a piece of music from a large collection based on the musical properties of that piece (e.g. "find all songs in my collection where the piano uses a ii-V-I chord progression in the chorus"). Music recommender systems are supported by similar technologies to find the songs that you like. Music analysis systems could furthermore allow to isolate that one amazing guitar riff from a music recording wherein many other instruments are playing at the same time, or to automatically create a karaoke version of that new pop song. These technologies can also be used to create and modify music in ways that were not possible before. Music generation software can assist a user in composing their own music even if they do not have a strong musical background. This could be useful, for example, to generate the accompaniment for a song written for a friend's birthday, or compose background music for an indie film where the creators do not have the budget to hire a professional composer. Music restoration software could transform old music recordings to sound as if they were recorded and produced according to contemporary standards. The possibilities are truly endless.

Of course, solving these complex challenges requires sophisticated computational methods. The development of these technologies is the mission of the field of Music Information Retrieval (MIR). As the name suggests, a considerable section of this field aims to extract human-interpretable information from music signals, ranging

from automatic genre classification to automatically transcribing particular instruments in a recording. However, research tasks that are not strictly about extracting (human-interpretable) information from music, such as music generation, are also within the broader scope of the MIR field.

This thesis addresses certain challenges within the MIR subfield of (audio-)content-based MIR, i.e. where the audio signal itself is the only source of information, instead of for example contextual information such as listening behavior statistics, music metadata, or a symbolic representation of the music such as sheet music or digital MIDI messages. This thesis is split into two parts. Part I discusses my contributions in the domain of automatic drum transcription, which aims to extract a symbolic transcription of the percussive parts of a music recording. In Part II, two systems are presented that use various MIR technologies to process and transform drum'n'bass music, a specific genre of electronic music, and as such automatically create new compositions.

Part I presents my contributions in the field of automatic drum transcription (ADT). I focus more specifically on improving the non-negative matrix factor deconvolution (NMFD) algorithm for the automatic decomposition of percussive solos. NMFD describes a mixture as the superposition of several sound sources, where each source is represented by a short spectrogram extract or "template" and a series of activations through time. The templates describe the sound that the corresponding source makes, i.e. in the context of ADT, each template captures a hit on a percussive instrument. The activations then indicate when and at which volumes a source is heard in the recording.

In this thesis, two issues are addressed that occur when NMFD is used for ADT. Firstly, the activations are often not "impulse-like": they do not localize the occurrence of drum sounds in a decomposed mixture at discrete time points, but they are instead

spread out over time. This does not make sense in the context of a drum recording, where it would be expected that the activation of a percussive sound is instantaneous. It also makes the decompositions unclear and less useful. In Chapter 3, this issue is addressed by requiring that the activations are binary: either an instrument is hit, i.e. the corresponding activation value is 1, or it is not hit, i.e. activation value 0, and small and unclear activation values are discouraged. This binary behavior is obtained by rewriting the activations as the output of a sigmoidal function, multiplied with a per-component amplitude factor. A regularization term then biases the decomposition to solutions with saturated activations. As demonstrated on a publicly available large-scale database of percussive recordings, this solution effectively leads to sparse and impulse-like activations and consequently to more interpretable decompositions.

The second issue is that NMFD has a tendency to capture sequences of drum strokes in a template, instead of a single drum hit, when the templates are relatively long. A naive solution is to simply use shorter templates. This is often not possible, though, as a recording might contain some drum sounds that have a relatively long decay time and therefore require long templates in order to be modeled appropriately. Instead, in Chapter 4, an ad-hoc modification to the NMFD update procedure is proposed that detects the emergence of secondary onsets in the templates during optimization, and that overwrites any excess drum hits in a template as soon as they start to appear. I illustrate that this indeed reduces the number of "double hits" in the discovered templates.

Part II of this thesis disseminates the results of two systems that use music processing technology to analyze and transform drum'n'bass (D'n'B) music.

Chapter 5 describes the implementation of an automated DJ system for D'n'B. In order to realize this system, a myriad of MIR techniques is combined. Firstly, the structure of the song is ana-

lyzed by means of tempo estimation, beat and downbeat tracking, and structural segmentation, which allows the DJ system to appropriately align the tracks in the DJ mix. These techniques exploit certain musical properties of D'n'B, which makes the system able to provide correct structural annotations for 91% of the tracks within the test set. This DJ system implementation takes common DJing best practices into account, such as phrasing (i.e., aligning tracks at segment boundaries for a structurally coherent mix), harmonic mixing (i.e., only mixing tracks that are in the same key or that are in keys related to each other according to the circle of fifths), avoiding vocal clashes (by means of a vocal detection classifier), and establishing a coherent and gradually evolving atmosphere throughout the mix (by embedding each song in a "genre space" using PCA). The system furthermore implements three types of transitions, resulting in a varied and enjoyable seamless DJ mix.

Chapter 6 demonstrates the application of an unpaired image-to-image translation technique to music spectrograms in order to realize a style transfer system between two subgenres of D'n'B. This technology is already used, for example, to transform photographs of landscapes into a (picture of a) painting in the style of a certain painter, or to transform images of zebra into images of horses and back. This chapter illustrates that this image-to-image translation network learns meaningful audio transformations when applied to music spectrograms. In this case, the system learns to modify the timbre of the drum hits in order to resemble the typical percussive timbre of the considered D'n'B subgenres. This is furthermore possible with a relatively small dataset (120 extracts of just over 5 seconds of audio for each subgenre).

Finally, Chapter 7 concludes this thesis by summarizing the obtained results and by giving an overview of potentially interesting avenues for further research. In short, in terms of using NMFD for ADT, it appears that NMFD is in some aspects too flexible, while in others it is too rigid. Therefore, the NMFD framework should

on the one hand become more constrained by incorporating musical knowledge about what constitutes a musically meaningful spectrogram template. On the other hand, NMFD should become more flexible, for example by allowing slight variations between different instantiations of the same template and also by allowing to perform only a partial decomposition of the spectrogram. Regarding ADT research in general, there is a positive evolution towards training systems on larger and more varied datasets. However, the MIR community should be careful not to try to create one-size-fits-all pre-trained solutions. Instead, there should also be continued research towards flexible drum transcription systems, for application scenarios where the percussive sounds do not fit within the predefined categories in the used training datasets or where little data is available to fine-tune existing pretrained systems (or to train new systems from scratch). Finally, systems that automatically transform and create music are becoming more potent and numerous. In their further development, it will be essential to include musicians from all music cultures and subcultures and to ensure that these new technologies are accessible and easy to experiment with by artists. After all, this is what I think is one of the most important missions of the MIR community: to advance the frontiers of how we can make and experience music, by realizing advanced technologies in service of the creator, the listener, and the music.

Samenvatting

We leven in digitale tijden, en dit laat toe om enorm interessante computerapplicaties te realiseren die muziek analyseren en ons in staat stellen ermee te interageren. Een voorbeeld van zo een toepassing is het snel en gemakkelijk terugvinden van muziek in een grote muziekbibliotheek op basis van de muzikale eigenschappen van dat muziekstuk ("zoek alle nummers in mijn bibliotheek waar de piano een ii-V-I akkoordprogressie volgt tijdens het refrein"). Daarnaast worden aanbevelingssystemen ondersteund door gelijkaardige technologieën om de muziek te vinden die jij leuk vindt. Met technieken voor muziekanalyse kan je ook die fantastische gitaarsolo uit jouw favoriete nummer extraheren en alle andere instrumenten automatisch dempen, of je kan dit gebruiken om automatisch een karaokeversie van dat nieuwe pop-nummer te maken. Het wordt ook mogelijk om muziek te maken en te bewerken op manieren die tot nu toe niet mogelijk waren. Muziekgeneratiesoftware kan een gebruiker assisteren in het componeren van een eigen stuk, zelfs als die gebruiker geen muzikale achtergrond heeft. Dit kan bijvoorbeeld gebruikt worden om de muzikale begeleiding te genereren voor een lied geschreven voor de verjaardag van een vriend, of om achtergrondmuziek te componeren voor een onafhankelijke film (een "indiefilm") waar geen budget voorhanden is om een professionele componist in te huren. Muziekrestauratieprogramma's kunnen oude muziekopnames transformeren zodat deze zo helder en scherp klinken alsof ze opgenomen werden in moderne studio's en geproduceerd werden volgens hedendaagse standaarden. De mogelijkheden zijn werkelijk eindeloos.

Voor het oplossen van deze complexe uitdagingen zijn gesofisticeerde computationele methodes nodig. Het ontwikkelen van de algoritmes die dit doen, gebeurt in het domein van de muziekinformatieëxtractie, in het Engels *"Music Information Retrieval"* of MIR genoemd. Deze naam geeft aan dat er binnen dit veld een specifieke focus is om interpreteerbare informatie uit muzieksignalen te extraheren, gaande van het automatisch classificeren van het genre tot het automatisch transcriberen van een specifiek instrument. Het is echter ook zo dat onderzoekstaken die strikgenomen niet gaan over het extraheren van (interpreteerbare) informatie, zoals bijvoorbeeld muziekgeneratie, ook tot het gebied van de MIR gerekend worden.

In deze thesis worden verschillende problemen bekeken die tot het domein van de (audio)inhoudsgebaseerde MIR behoren. Dit wil zeggen dat het audiosignaal zelf de enige bron van informatie is en dat er geen andere informatiebronnen bekeken worden, zoals statistieken over het luistergedrag van individuen of groepen luisteraars, metadata zoals de artiest, het land van oorsprong of het jaar van uitgave van de muziek, of een symbolische voorstelling van de muziek (bijvoorbeeld een partituur of digitale MIDI-signalen). De thesis is opgesplitst in twee delen. Deel I bespreekt twee bijdragen in het veld van de automatische drumtranscriptie, waar het doel is om een symbolische voorstelling van de percussieve delen van de muziek te extraheren uit de audio. In deel II worden twee systemen besproken die verschillende MIR-technologieën gebruiken om drum'n'bass-muziek, een specifiek genre van elektronische muziek, automatisch te verwerken en om te zetten in nieuwe composities.

Deel I bespreekt twee bijdragen in het veld van de automatische drumtranscriptie (ADT). Daarbij ligt de focus specifiek op het verbeteren van het niet-negatieve matrixfactordeconvolutie-algoritme (NMFD) voor het automatisch ontleden van percussieve opnames. NMFD beschrijft een audiosignaal (voorgesteld als een spectrogram) als de superpositie van verschillende geluidsbronnen, waar-

bij iedere bron beschreven wordt aan de hand van een kort spectrogramextract of "sjabloon" en een reeks van activaties doorheen de tijd. De sjablonen beschrijven welk geluid de bijhorende bron maakt; in het geval van ADT is dit een slag op een percussief instrument. De activaties duiden de momenten aan waar en het volume waarmee deze geluiden voorkomen in de opname.

In deze thesis worden twee problemen aangepakt die optreden als NMFD gebruikt wordt voor ADT. Ten eerste zijn de activaties vaak geen impulsen, wat wil zeggen dat ze de drumgeluiden in een ontbonden geluidsfragment niet lokaliseren op discrete tijdspunten. In plaats daarvan zijn de activaties eerder uitgespreid in de tijd. Dit klopt niet in de context van percussieve geluiden, waar een ogenblikkelijke activatie van een drumgeluid verwacht zou worden. Dit maakt de ontbindingen bovendien onduidelijk en minder bruikbaar. In hoofdstuk 3 wordt dit probleem aangepakt door te eisen dat de activaties binair moeten zijn: ofwel wordt een instrument aangeslagen, wat overeenkomt met een activatiewaarde van 1, ofwel is dat niet zo, wat betekent dat de activatie 0 is. Kleine en onduidelijke activatiewaarden worden op die manier vermeden. Dit binair gedrag wordt gerealiseerd door de activaties te definiëren als de uitkomst van een sigmoïdale functie, vermenigvuldigd met een amplitudefactor per component. Een regularizatieterm zorgt ervoor dat het ontbindingsproces oplossingen verkiest waarbij de activaties in het gesatureerde regime van de sigmoïdale functie liggen. Met experimenten op een publieke grootschalige dataset van percussieve opnames wordt aangetoond dat deze oplossing effectief leidt tot dunbezette en impulsachtige oplossingen and bijgevolg tot meer interpreteerbare ontbindingen.

Het tweede probleem is dat NMFD de neiging heeft om opeenvolgingen van drumgeluiden te vatten binnen een sjabloon, in plaats van een enkel drumgeluid, wanneer de sjabloonlengte relatief groot is. Een naïeve oplossing is het verkleinen van de sjabloonlengte, maar vaak volstaat dit niet. Sommige opnames bevatten bijvoor-

beeld drumgeluiden met een relatief trage daling in volume (in Engels vakjargon gekend als de "*decay*") en die moeten bijgevolg gemodelleerd worden met een lang sjabloon. Hoofdstuk 4 bespreekt een ad-hoc aanpassing van het optimalisatieproces van NMFD waarbij het opkomen van meerdere drumgeluiden binnen een sjabloon gedetecteerd wordt. Van zodra een extra drumslag voorkomt binnen een sjabloon, dan wordt deze overschreven. Experimenten tonen aan dat dit effectief het aantal dubbele drumgeluiden binnen de sjablonen vermindert.

Deel II bespreekt twee systemen die muziekverwerkingstechnologieën gebruiken voor het automatisch bewerken van drum'n'bass (D'n'B), een sub-genre van de elektronische dansmuziek.

Hoofdstuk 5 beschrijft de implementatie van een automatisch DJ-systeem voor D'n'B. Dit systeem bestaat uit een combinatie van verschillende MIR-technieken. Zo wordt de structuur van de muziek geanalyseerd aan de hand van technieken zoals tempobepaling, ritme- en opmaatdetectie, en structurele segmentatie. Dit laat toe om de verschillende nummers in de DJ-mix correct met elkaar te aligneren. Deze technieken buiten bepaalde eigenschappen van D'n'B uit, wat ervoor zorgt dat het systeem tot 91% van de nummers in de testverzameling correct kan annoteren. Het DJ-systeem houdt bovendien rekening met technieken die typisch gebruikt worden door DJs, zoals frasering (waarbij de muzikale structuren in de verschillende nummers met elkaar gesynchroniseerd worden om te zorgen voor een structureel coherente mix), harmonisch mixen (zodat enkel nummers tegelijk afgespeeld worden als die in dezelfde toonaard of in gerelateerde toonaarden volgens de kwintencirkel staan), het vermijden van overlappende zang in verschillende nummers, en het creëren van een coherente en evoluerende sfeer doorheen de mix (door de muzieknummers onder te brengen in een "genreruimte" met behulp van PCA). Het systeem kan drie soorten overgangen uitvoeren, wat zorgt voor een gevarieerde en aangename DJ-mix.

Hoofdstuk 6 demonstreert een stijloverdrachtssysteem tussen twee sub-genres van D'n'B. Dit wordt gerealiseerd door de toepassing op muziekspectrogrammen van een zelflerende techniek die afbeeldingen van het ene domein naar het andere omzet. Deze technologie wordt bijvoorbeeld al gebruikt om foto's van landschappen te transformeren tot (een afbeelding van) een schilderij in de stijl van een bepaalde schilder, of om afbeeldingen van paarden om te zetten in afbeeldingen van zebra's en omgekeerd. Dit hoofdstuk illustreert dat deze afbeeldingvertalingstechniek betekenisvolle audiotransformaties leert wanneer die toegepast wordt op muziekspectrogrammen. In dit geval leert het systeem om het timbre van de drums aan te passen zodat dit overeenkomt met het typische percussieve timbre in de gebruikte D'n'B sub-genres. Dit is bovendien mogelijk met een relatief kleine dataset (120 audiofragmenten van elk ongeveer 5 seconden lang voor elk sub-genre).

Hoofdstuk 7 sluit deze thesis af met een overzicht van de belangrijkste onderzoeksresultaten en met een bespreking van interessante pistes voor verder onderzoek. Samengevat worden de volgende openstaande problemen besproken. In de context van NMFD voor ADT lijkt het erop dat NMFD enerzijds te flexibel is en anderzijds net te rigide. Een mogelijke piste is dus om het NMFD-algoritme meer te beperken door muzikale kennis in te bouwen over hoe een muzikaal betekenisvol sjabloon er precies uit moet zien. Anderzijds kan er onderzocht worden hoe NMFD meer flexibel gemaakt kan worden, bijvoorbeeld door toe te laten dat de verschillende instantiaties van de sjablonen doorheen het spectrogram licht variëren, en ook door het algoritme aan te passen zodat ook gedeeltelijke transcripties van een spectrogram mogelijk worden. In het algemeen is er in het ADT-domein een trend naar het trainen van *machine learning*-systemen op grotere en meer gevarieerde datasets, wat een positieve evolutie is. Desalniettemin moet de MIR-gemeenschap er ook aandachtig voor zijn om geen vooraf getrainde "one-size-fits-all"-oplossingen te proberen te maken.

Er moet nog voldoende onderzoeksaandacht gaan naar flexibele transcriptiesystemen, voor applicaties waar percussieve geluiden niet binnen de vooraf gedefinieerde categorieën van de gebruikte trainingsdatasets vallen of waar weinig data voorhanden is om vooraf getrainde systemen af te stellen (of vanaf nul te trainen). Ten slotte worden de systemen om automatisch muziek te maken en transformeren steeds krachtiger en talrijker. In hun verdere ontwikkeling is het belangrijk om muzikanten vanuit alle muziekculturen en -subculturen te betrekken, om ervoor te zorgen dat deze nieuwe technologieën toegankelijk zijn en dat er gemakkelijk mee geëxperimenteerd kan worden door artiesten. Dit is volgens mij immers een van de belangrijkste missies van de MIR-gemeenschap: de grenzen verleggen van hoe we muziek maken en ervaren, door het realiseren van geavanceerde technologieën in dienst van de maker, de luisteraar en de muziek.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADT	Automatic Drum Transcription
AMT	Automatic Music Transcription
BPM	Beats Per Minute
CQT	Constant Q-Transform
DFT	Discrete Fourier Transform
DJ	Disk Jockey
D'n'B	Drum'n'bass
DTW	Dynamic Time Warping
EDM	Electronic Dance Music
FFT	Fast Fourier Transform
GAN	Generative Adversarial Network
HWR	Half-wave rectification
IBI	Inter-beat interval
KL (divergence)	Kullback–Leibler (divergence)
MAE	Mean Absolute Error
MFCC	Mel-frequency cepstral coefficient
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
NMF	Non-negative matrix factorization
NMFD	Non-negative matrix factor deconvolution
ODF	Onset Detection Function
PCA	Principal Component Analysis
RMS	Root Mean Square
SSM	Self-similarity matrix
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine

1 Introduction

Music is all around us: we listen to it at home, it's broadcast on the radio, there's music in TV commercials and in the background in a café, we're dancing to it at festivals. We often interact with our music digitally: music is stored on a computer or played through a streaming app on your phone, the DJ carries it with them on a USB stick to the club. However, we might want to interact with our music and our music collection in ways that are, in fact, not at all trivial. Let us consider some examples of such interactions, and what kind of musical understanding is expected in these examples from the software on our computer or smartphone.

For example, let's imagine that you're hosting a party tonight. Your friends are invited, the food and drinks are ready to be served, and of course, there will be music. As the host, you would like to prepare a playlist for the night. During your preparations, you can think of a jazzy song that you would like to add to it... but you forgot what it's called! Luckily, you remember what the pianist

plays in one part of the song, and in your (very advanced) music streaming app, you type: "find all jazzy songs in my collection where the piano plays a ii-V-I chord progression in the chorus.".

In order to find the songs that match this query, the music streaming software needs to have an understanding of many aspects of the songs in the music collection. Firstly, it needs to be able to classify songs according to their genre and to detect nuances and influences from other genres in the music in order to find all jazz (and "jazzy") songs in the catalogue. It then needs to recognize the sound of a piano and discern it from other similarly sounding instruments. This instrument recognition needs to happen for music recordings where many other instruments are playing at the same time, which makes it even more difficult. The software furthermore needs to not only recognize the piano, but also needs to analyze what notes are being played on it – and determine what musical key the music is in – to figure out what chords are being played. Finally, it needs to know where the chorus is in the music: this requires an understanding of the structure of a musical piece, knowing what musical cues are relevant to detect the chorus, and the ability to identify these musical cues in the audio signal itself.

Let's consider a second example: an amateur rock band is coming together for band practice and a subsequent jam session. Unfortunately, their singer is still too hoarse from the last gig and cannot participate. To solve this, the other band members would like to practice alongside an "*a capella*" version of the song that they are covering, but they cannot find one: only a "full" recording exists, with vocals and backing instruments. Would there be a way to isolate only the vocals from an existing music recording and filter out all other instruments?

Afterwards, during the jam session, the drummer is looking for inspiration to spice up his rhythms. What if a computer program could generate drum pattern variations based on what the drummer and the other band members have been playing?

These are examples of music manipulations that require a fine-

grained understanding and ability to modify information in the music signal.

These examples show that it would be incredibly interesting to extract all kinds of information from music recordings. To find songs in a music streaming app based on their musical properties, a music streaming service could hire expert musicologists to analyze and annotate all songs in a music catalogue with this kind of information. However, this approach quickly becomes infeasible for large collections of music where tens of thousands of new songs are added every day. Automatic music generation and decomposition are furthermore inconceivable without computer techniques. Therefore, automated approaches for musical audio analysis are needed.

This thesis is situated in the scientific domain of music information retrieval (MIR), which concerns itself with addressing challenges like the ones mentioned in this introduction. Section 1.1 gives an introduction of how MIR techniques work in practice, and it provides a high-level overview of the cross-section of the MIR field that is most relevant to this work. This thesis focuses on the one hand on the decomposition of percussive music recordings, and on the other hand on transforming drum'n'bass music using MIR techniques in order to create new compositions. Section 1.2 explains the musical perspective that inspired me to perform this research, and as such provides a musical context in which this thesis can be understood. Section 1.3 then concludes this chapter by giving an overview of the contributions of this thesis.

1.1 Music Information Retrieval

The previous section demonstrates that music is an incredibly rich signal that contains a lot of information. It would hence be very interesting to have methods to extract, analyze and transform this information. The field of music information retrieval (MIR) is the research domain that concerns itself with developing computational

methods to process music and data about music. This data can be an audio recording of the music, but also metadata such as information about the artist, the country of origin of the music, listening patterns of individuals or groups, music playlists etc. Music can also be represented in symbolic formats, e.g. as sheet music or MIDI data. The field of MIR lies on the intersection of musicology, signal processing, natural language processing and machine learning, and it is therefore incredibly rich and broad.

Of course, extracting information from music is usually done in order to solve a particular problem or create a particular application. The introduction of this chapter already gives an overview of some typical end goals for MIR: facilitating music retrieval and browsing through music libraries based on a myriad of musical features such as timbre, groove, rhythm, tempo, instrumentation etc.; music recommendation and playlist generation; automatic synthesis of new music; automated music analysis etc. Other applications include automatic score following, automatic music transcription, music database organization, audio restoration and many more.

The following sections serve as a brief introduction into the field of MIR. Section 1.1.1 briefly covers how musical audio is often represented as a *spectrogram*, i.e. a two-dimensional time-frequency representation of the audio. Section 1.1.2 then illustrates how information is usually extracted from such a digital representation, i.e. either by calculating "engineered" features or by using algorithms that automatically learn to define and extract their own features. Finally, Section 1.1.3 gives an overview of recent advances in MIR sub-domains that are of particular interest to this thesis, and hence provides a view on the broader context in which this thesis is situated. For a more in-depth overview of MIR techniques for music audio analysis, I refer to the book by Müller [1] or the book by Knees and Schedl [2].

1.1.1 Digital representations of a music signal

In order for a computer system to process an audio signal, it needs to have a digital representation of that audio signal to work with. The most common digital representation of audio signals is the waveform, which is a discrete sequence of values representing the evolution of the (quantized) amplitude of the audio through time, sampled at a specific sampling rate. It is often quite difficult, though, to extract musical information from this waveform representation directly: while the information of interest is present in the audio signal, it is typically entangled with other information in the audio. To solve this, the audio is often transformed into a so-called *time-frequency* representation, wherein the relevant information is more discernible. From the audio data – or a transformation thereof – musical features can then be extracted. These features can be designed manually – a practice called "feature engineering" – or they can be transformations learned from data by a parameterized model using machine learning or deep learning – this would be called "artificial intelligence" in layman terms.

Time-frequency representation of audio signals

A common type of transformations that are used in many MIR applications (and more generally in many audio and signal processing applications) are *time-frequency* transformations. These transform the one-dimensional audio signal into a two-dimensional view of the audio content, showing the evolution of the audio through both time and frequency. This allows to more easily discern distinct and simultaneous events in the audio.

The audio waveform describes the evolution of the amplitude of the audio through time. However, one can also describe the evolution of the amplitude through frequency. For example, in music, sounds range from low-frequency bass notes and singing in the middle frequencies to high-frequency noise, whistles etc. The frequency content of the audio – i.e. the distribution of the energy through

frequency – can be analyzed by applying the discrete Fourier transform (DFT). This is a mathematical operation that transforms a discrete time signal of finite length – such as an audio waveform – into an array of complex Fourier coefficients. These coefficients describe the analyzed signal as the weighted sum of (complex) sinusoids of increasing frequencies. As such, the DFT provides an analysis of how strongly these frequencies are present in that signal. However, like the waveform only describes the temporal behavior of the audio, the DFT only describes its behavior in the frequency domain. It would often be more useful to have a view of how the audio amplitude or energy evolves through both time and frequency.

This is the purpose of the short-time Fourier transform. As its name suggests, the STFT calculates the Fourier transform of short time snippets of the input (audio) signal. The STFT is calculated by first slicing the signal into short, consecutive, overlapping frames of equal length, typically of a few tens of milliseconds long. Then, the DFT of each time slice is calculated, thereby providing an analysis of what the frequency content of the audio signal looks like at each moment in time. Because many short consecutive frames are analyzed, one obtains a view of the evolution of the audio signal through both time and frequency. The short audio frames are usually multiplied by a (typically bell-shaped) windowing function to smooth out the edges of these frames, in order to reduce the impact of artifacts in the frequency analysis. Successive frames overlap partially in order to ensure that no information is lost throughout this process of windowing the frames, making the resulting transformation invertible (i.e. one can go back from the STFT to the waveform representation when the amount of overlap is chosen properly in accordance with the windowing function).

The outcome of the DFT, and therefore of the STFT, is complex valued: the amplitude of a Fourier coefficient represents how strongly the corresponding sinusoid is present in that slice of audio, whereas the phase information describes how the corresponding sinusoid is shifted with respect to the start of the analysis frame. The

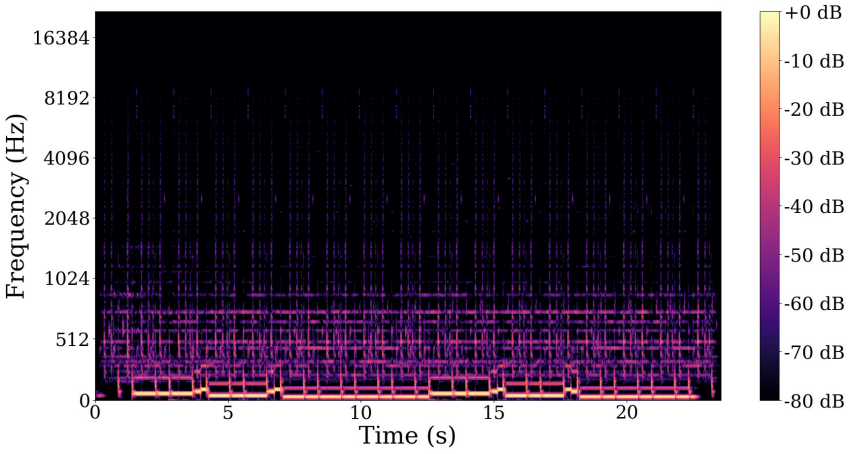


Figure 1.1: An example of a music spectrogram with a logarithmic amplitude scale and a Mel-scaled frequency axis. One can discern distinct musical events, such as percussive hits (long vertical lines), and individual bass notes (yellow horizontal lines at the bottom).

frequencies represented by the Fourier coefficients are distributed evenly between 0 Hz and the Nyquist frequency $f_s/2$, i.e. the highest frequency that can be faithfully captured in a digital signal sampled at a sampling rate f_s (without introducing aliasing). In many applications, only the magnitudes of the STFT coefficients are considered and the phase information is discarded, typically because the phase information is hard to model. This results in a so-called STFT "spectrogram" representation of the audio, which shows the evolution of the strength of the audio content through time. Note that discarding the phase information makes the resulting representation non-invertible, i.e. one cannot go back to the audio domain anymore. An example of a spectrogram is shown in Figure 1.1.

When calculating the STFT, a trade-off is made between the time resolution and the frequency resolution of the resulting time-frequency representation. This is because the number of frequencies

that one can 'read out' on the frequency axis is determined by the length of the analysis frames: using analysis frames of N samples leads to $N/2$ "frequency bins" in the corresponding FFT[1]. Hence, using longer frames leads to a better frequency resolution: since there are more frequency bins evenly distributed between 0 Hz and the Nyquist frequency $f_s/2$, one gets a finer-grained view of the frequency content of that audio frame. However, this comes at the cost of a reduced time resolution: using longer frames makes it more difficult to discern when exactly certain events occur in the audio.

The spectrogram representation described so far uses a linear frequency axis and a linear amplitude scale. This is not in accordance with the non-linearities in human hearing, though. Consequently, the STFT spectrogram is often processed further in order to correspond better with how humans perceive sound. Firstly, the spectrogram amplitudes are often transformed into a logarithmic (decibel) scale. Secondly, spectrograms often have a non-linear frequency axis in order to better correspond with the non-linear perception of frequency [3]. Examples include the Mel-scaled spectrogram and the Constant-Q Transform [4].

1.1.2 Extracting information from music audio

Mid- and high-level features of audio signals

Spectrograms provide an alternative view of the audio signal and make musical events in the audio easier to discern. This allows to build algorithms that detect these events in the spectrogram. However, there are many other types of descriptors or *features* that can be extracted from the audio signal.

[1] The discrete Fourier transform of a signal of length N results in N Fourier coefficients. However, since the audio signal is a real-valued signal, only half of them are useful: the last half of the coefficients are the complex conjugate of the first half in reverse order.

The ultimate goal of content-based MIR is often to extract some human-interpretable information from the audio signal. This information can be high-level and abstract; therefore, features that capture this type of information directly are called *high-level* features. Examples include e.g. the genre of a piece of music, the instrumentation that is used in a recording, the emotion that a piece of music evokes, or the use of certain playing techniques (e.g. "legato"). These features describe human-interpretable concepts, but are sometimes ambiguous or vague and therefore hard to formalize (i.e. there is a large semantic gap, as is explained below). Other features capture information that is more closely related to the audio signal itself. Examples include the energy of the audio through time, the number of zero-crossings of the audio signal, the loudest frequency bin of the spectrogram at each point in time etc. These features are usually easily defined and formalized, but they are often not directly linked to interpretable or meaningful musical descriptors that one may want to extract. Finally, in between low-level and high-level features are so-called *mid-level features*. These often combine or transform low-level descriptors in such a way that they become better correlated with certain high-level musical concepts, while not being directly interpretable themselves. One example would be the Mel-frequency Cepstrum Coefficients (MFCCs), which capture timbrical properties of the audio [1]. Note that the distinction between low-level, mid-level or high-level features is in fact a continuum, and there are no clearly defined borders between these categories.

Features that quantify a musical property require some abstraction of that musical concept in order to be implemented into a computer algorithm. For example, imagine that one would want to ranks songs by how "energetic" or "exciting" they are. Formalizing this into a computer system requires defining what "energetic" means and how it is calculated, even though this is generally not clearly defined or may depend on many aspects that in themselves are hard to measure. This leads to a phenomenon known as the

"semantic gap", i.e. there is a discrepancy between the (potentially ambiguous or vague) desired meaning of a feature and the formal implementation of that feature. This implies that in the design of MIR systems, it is important to check with the users whether the implemented features capture the desired musical concepts well. Knees and Schedl [2] argue that there might even be a semantic gap between people, meaning that different people might have a different understanding of what a certain feature means and how it should be implemented. In order to further narrow the semantic gap, they argue that it is important to investigate how music perception is influenced by differences in background or culture, and consequently tailor computational models to the actual music perception and the specific requirements of the users.

Learning features from data

The features mentioned in the previous section are typically designed "by hand" in order to extract the information from the audio signal that is deemed relevant and informative for the problem at hand. There are, however, several drawbacks to this approach of "feature engineering". Firstly, it is time-consuming and requires significant intellectual labor. It also requires a high level of expertise and a thorough analysis of the problem at hand, in order to ensure that the semantic gap between musical concept and implemented feature is as small as possible. However, the features that could be most informative might be quite elusive, i.e. it might not be obvious how existing low- and mid-level features should be combined to capture the desired musical property.

Another popular paradigm is therefore to *learn* features from data. Deep neural networks are a type of machine learning models that are commonly used for this task. The following section aims to provide a high-level overview of how these techniques work at a glance, without going into deep technical detail. I refer to Goodfellow et al. [5] for an in-depth discussion of deep learning and

representation learning using deep neural networks.

A deep neural network is a parameterized mathematical model that describes the transformation of some input data – e.g. an image – into an output – e.g. the probability that a certain object is visible in that image. Such a network consists out of a concatenation of multiple "layers" that each represent a mathematical operation: each layer takes the output of the preceding layer as its input, transforms it, and then passes the transformed output on to the next layer. The outputs of a layer represent a transformed and often more compact view of the input data, i.e. they can be interpreted as features of the input data. Every layer has a set of associated parameters that define the specific transformation that it applies.

In order to learn meaningful transformations (and, consequently, representations), the parameters of the deep neural network are "trained" on a large collection of data in order to establish a desired input-output behavior. This behavior is quantified by means of a "cost function", which describes how well the model is performing on the imposed learning task. There are two major paradigms for learning representations: supervised and unsupervised learning. In supervised learning, each data point in the training dataset is paired with a desired output. In unsupervised learning, data points are not labeled, and the deep neural network learns representations using only this unlabeled data. In between unsupervised and supervised learning is semi-supervised learning, where only a fraction of the data is labeled.

The algorithm to train a neural network operates as follows. First, the model is applied to one or several input examples, which produces corresponding outputs. The cost function then quantifies the performance of the model on the provided inputs, e.g. by measuring the discrepancy between actual and expected output in the case of supervised learning, or by measuring the ability of the model to reconstruct the input in the case of a so-called "autoencoder" for unsupervised learning. Because this cost function is chosen to be

differentiable with respect to the neural network parameters, one can determine the direction in which the parameters should be updated so that a lower value of the cost function is expected the next time the model is applied to the data. The parameters are then changed accordingly, i.e. by applying the gradient descent optimization algorithm (or a variant thereof). These steps are performed iteratively until the model converges to a minimum of the cost function. Throughout this process, the model learns an internal feature representation of the input data. These internal features can sometimes be reused later in other learning tasks, which is a technique known as "transfer learning".

As an example to make this more concrete, consider a deep neural network model that is trained supervisedly in order to perform a music tagging task [6]. This is done by iteratively providing several (Mel-scaled) spectrograms to the neural network. Each input spectrogram in the training dataset is paired with a 50-dimensional binary vector, which describes whether or not a certain tag (e.g. genre, era, instrumentation) out of a pool of 50 fixed tags is relevant for that piece of music. For each example, the neural network produces an output that also is a 50-dimensional vector. The discrepancy between the target binary vector and the actual output vector is calculated by means of the binary cross-entropy cost function, and the parameters of the neural network are then optimized by means of stochastic gradient descent.

Throughout this process, the model learns to define its own features. The structure of stacked convolutional layers that is used in the network in this example is expected to learn hierarchical time-frequency patterns: the lowest convolutional layers learn fine-grained patterns, while the layers at the end of the chain learn coarse patterns. The work considered in this example [6] illustrates that such a network learns meaningful features that outperform baseline models using engineered features (MFCCs) and that can furthermore be transfered to other MIR tasks as well.

1.1.3 MIR tasks of interest

In the following, I provide a high-level overview of several MIR tasks that are particularly relevant to this thesis. Three MIR sub-fields are highlighted: automatic music transcription, automatic music decomposition and sound source separation, and automatic music generation and (creative) modification.

Automatic music transcription

The goal of automatic music transcription (AMT) is to extract a symbolic representation from the audio recording of a musical performance. This symbolic representation can be in the form of the sheet music for that piece, a stream of MIDI messages, a simplified "piano roll" representation or any similar discrete representation of the musical events in the audio. Such a transcription may or may not include information about the expressive performance, such as velocity information, the use of techniques such as vibrato on a violin or note bending on a guitar etc. These systems are developed to transcribe both solo recordings of the considered instrument, as well as to transcribe the instrument of interest in the presence of other instruments in the recording. Transcription in a polytimbral setting – i.e. where multiple instruments are playing together – is often considered more difficult, because the other sounds in the recording could obscure the instrument of interest, or they could mistakenly be recognized as coming from that instrument. Transcription systems are also often tailored to transcribe a specific instrument. For example, this thesis considers the transcription of percussive instruments, as explained in Section 1.3.

A robust and flexible AMT system is incredibly useful for many applications. For example, it acts as a building block of content-based searching through a music database based on a melody or rhythm in the music. It furthermore allows an automated musicological analysis of a music piece and offers insight into how a music piece is composed or what expressive performance techniques are

used in a particular recording. Real-time transcription in turn enables applications that react to the music while it is being played, such as generative music systems (see below) that play along with a human artist or advanced visual and lighting effects at a festival.

High-quality transcription technologies are also valuable from a research standpoint. Much MIR research relies on the availability of large volumes of annotated data, for example for training machine learning models and for evaluation purposes. However, creating these datasets is a time-consuming and laborious task. Automatic transcription systems can alleviate this issue somewhat by annotating large volumes of data automatically. They could hence be used, for example, to pre-train deep learning models [7], to annotate large datasets in scenarios where the annotation quality does not need to be perfect, or as a tool to quickly annotate the easiest examples, so that an expert annotator then only needs to verify these and spend time on manually annotating the hardest examples.

Musical audio decomposition and sound source separation

The goal of sound source separation is to dissect an audio recording into several audio streams that represent the different sources that together make up that recording. In the context of music, these sources can be different things, depending on the application. A common and intuitive use case is the creation of karaoke versions of a song. The two different sources are then the (lead) vocals on the one hand and "everything else" in the audio recording on the other hand. One can go further than that, though: a sound separation system could extract several instrument categories from a recording at once. For example, it could separate a music recording into the percussion (drums), the melodic instruments, the bass, and the vocals, resulting in different audio streams for each instrument category (the individual audio streams of each instrument – or sometimes of groups of related instruments – are called the "stems" in a music production context).

Recently, incredible amounts of progress have been made in music source separation quality. One commonly used contemporary technology is U-Net [8], a specific deep neural network architecture that achieves great results in terms of separation quality [9]. Its success even led to commercial adoption, for example in audio editing and mixing software or in software for DJing [10]. Some pre-trained source separation models have also been made open source [11, 12]: even though they were trained on proprietary data, the resulting models can still be used for free.

Isolating individual sound sources from a music mixture facilitates various interesting applications. For example, as with AMT, it allows to analyze music production methods, gain insight into expressive performance techniques, better observe the sound design in a piece and so on. It also allows to repurpose particular elements from a music recording, e.g. by generating *a capella* or karaoke versions of existing songs, or by sampling particular sounds to reuse in new compositions. These applications are not new; however, they typically require access to the stems used in the production of the song, or they require extensive filtering and other sound modifications of the original recording, which potentially leads to an inferior sound quality for the resampled material.

Finally, like AMT systems, separation systems can provide a solid foundation to build other MIR systems upon. For example, they enable a "divide-and-conquer" strategy for the automatic transcription of polytimbral music, where a source of interest is first separated from the polytimbral recording and then transcribed with a monophonic music transcription system.

Limitations of automatic music transcription and music sound source separation systems

A drawback of the aforementioned systems (and of the publicly

available large scale datasets that are used to develop these systems) is that they usually consider pre-defined and generalized instrument classes to transcribe or extract from a music recording. For example, both the pre-trained Spleeter model [11] and the Open-Unmix model [12] separate a music recording into the stems "vocals", "drums", "bass", and "other"; Spleeter also has configuration that allows to extract all of these plus an individual "piano" stem. While this already allows for many interesting applications, many other use cases do not fit within this musical paradigm. This could for example be the case for e.g. non-Western music, which would consider completely different types or groupings of instrumentation. Another example would be certain genres of electronic music, where multiple layers of synthesized sounds can be present which might not belong uniquely to any of the aforementioned instrument groupings. A similar argument holds for music transcription systems. For example, automatic drum transcription systems are often designed to transcribe certain types of drum events, e.g. the kick drum, snare drum, hi-hat, and others. The datasets on which these systems are trained define – at the time of implementing the system – what types of sounds "belong" to each transcribed instrument class. However, certain recordings could contain sounds that fulfill one particular percussive role (e.g. that of the snare drum), but that sounds very different from a conventional snare drum. The transcription system might therefore not be able to transcribe such recording correctly, due to design decisions made for the transcription system and biases present in the training dataset.

Other use cases might benefit from higher granularity of decomposition. For example, a music producer might be interested in extracting not a stem of all the drums mixed together, but rather to isolate all individual drum instruments individually, thereby obtaining the isolated stems for the kick drum, snare drum, hi-hats, cymbals etc. The granularity of these decompositions can in turn depend on the use case. For example, should all hits on the hi-hat be grouped into one stem, or rather into several stems depending

on the playing technique (open hit, closed hit, pedal etc.)? Should all cymbals be grouped together? This dependence on the use case makes developing one-size-fits-all solutions all the more difficult.

Finally, developing and adapting the aforementioned transcription and sound source separation systems is often quite challenging. This is partially because they require large-scale datasets to be trained on. Therefore, research is needed into automatic music decomposition and transcription systems that do not depend on large datasets in order to be effective, or that can adapt to new use cases with little (additional) data.

Automatic music generation and modification

Computer algorithms can be used not only to analyze music, but also to create and modify music. Creative MIR – as in "MIR systems that create new music" – receives considerable attention within the academic MIR community. For example, at the 2019 edition of the International Society for Music Information Retrieval (ISMIR) conference, one of the major conferences within the field, more than one in eight papers (15 out of 114 papers) presented a novel technique, analysis or dataset related to automatic music generation[2]. At the 2020 edition of that conference this was more than one in six papers (22 papers out of a total of 116 papers). Their appeal also reaches a broader, non-academic audience. This is clear from initiatives such as the 2020 "A.I. Song Contest" by the Dutch broadcast service VPRO [13], the documentary for the program Pano by the Flemish public broadcast VRT on the automatic composition of piano music [14], and art projects where

[2] Note that, in order to obtain this count, I had to make a judgment of whether or not each paper was 'related' or 'relevant' to the topic of automatic music generation. Often this is quite clear, although sometimes it is not, so another observer might come to a slightly different count than me. Nevertheless, the number is still indicative of the popularity that the music generation topic has within the ISMIR community. Note that I only considered full papers and not e.g. late-breaking demos.

famous artists such as Stromae or Massive Attack experiment with A.I. technologies for their compositions [15, 16]. Finally, some of these generative MIR technologies have seen successful commercial adaptation [17, 18], again underscoring the artistic, scientific and commercial value of creative MIR.

Below, I give an overview of several automatic music generation techniques, to illustrate the capabilities of these systems. Creative MIR systems exist in many flavors and at different musical levels, ranging from generating individual melodies and rhythmic patterns to full songs. One can furthermore make a distinction between methods that generate music or musical elements in the symbolic domain and methods that generate music as a waveform or spectrogram.

Symbolic music generation automatically creates music as a stream of musical notes or MIDI messages. It thus acts at the compositional level; in order to obtain music that we can listen to from this symbolic representation, it needs to be synthesized after generation. One example of such a system is MuseNet, released by OpenAI in 2019 [19]. MuseNet generates musical compositions as MIDI in various musical styles, ranging from classical music to jazz and popular music (e.g in the style of The Beatles). This music is generated by giving the system an initial "prompt". MuseNet then completes the prompt by recursively predicting what the next token in the MIDI sequence could be, given all preceding tokens. This is realized by training a so-called *language model* on a large corpus of symbolic music in the aforementioned music styles.

MuseNet is an example of the automatic generation of full compositions. However, symbolic music generation is also used to create specific musical elements, or to modify existing compositions in an intelligent way. The plugins in the Magenta Studio bundle [20] are examples that do this. The bundle includes tools for the continuation of rhythms or melodies, for generating rhythms

or melodies (without any input), and for interpolating between the musical qualities of two input melodies or rhythms. It furthermore features a tool for humanizing drum loops by introducing velocity and timing variations, and a tool to create a drum loop conditioned on the rhythm of any input, which can be used to generate a percussive backing for a bassline or melody. Within generative MIR there also is a trend to make generative systems interactive [21–23], and as such give artists control over A.I.-generated music.

There also exist systems that generate music in the audio domain. As with symbolic music generation, a distinction can be made between systems that generate complete compositions and systems that generate only particular types of audio or music elements. These systems can also be categorized based on whether they generate audio as a waveform or as a spectrogram. Both approaches are notably difficult for their own reasons. For example, generating waveforms directly requires modeling very long sequences of waveform values, and most contemporary sequential models struggle with capturing long-term dependencies at this scale [24]. In this sense, generating spectrograms is easier, as the time resolution of a spectrogram is typically much larger than that of a waveform. However, obtaining realistic audio from a spectrogram requires modeling the phase of the (complex-valued) spectrogram, which is difficult to do [25]. Note that some approaches attempt to generate audio in other ways as well, for example by implementing common digital signal synthesis and processing methods in a differentiable manner and then learn how to control the audio generation parameters using a neural network [26].

An example of a (perhaps quite ambitious) project in this field is Jukebox by OpenAI [27]. Jukebox is a neural network that generates full compositions directly in the waveform domain. The model is able to generate multi-instrumental music in a variety of genres and artist styles, ranging from country, pop and rock to heavy metal and “classical pop”. Generating musical audio from scratch

is incredibly challenging, though [24]. For example, one limitation that Jukebox shares with many other music generation applications is the lack of long-term structure in the music [27].

There also are plenty of examples of audio and spectrogram generation tools that focus on particular compositional elements, as opposed to generating full compositions. Examples include systems for generating drum samples [28, 29] and for generating isolated sounds for various melodic instruments [25, 30]. Some of these systems are capable of transforming sounds from one "domain" to another [31], such as transforming a sung melody to sound as if it is played on a violin [26], or combining the sounds of a violin and the meowing of a cat [30].

Finally, creative MIR can go beyond the direct generation of music (either in a symbolic format, as a spectrogram or as a waveform). One interesting example is the automatic generation and interpolation between the parameters of software synthesizers [32]. Other works are transforming existing music and sound rather than generating new music. This includes systems that allow to transform one sound into another domain (as mentioned earlier), automatic mash-up and music mosaicing systems [33–35], automated DJ systems etc. While these technologies do not directly generate music, they still learn from and/or transform musical data and then allow artists and listeners to interact with music in novel ways.

1.2 A personal view on making and performing music with MIR technology

This thesis disseminates several technological contributions to the field of MIR. The previous section hence gives an overview of the technological and scientific context that the presented work resides in. Of course, there also is a strong musical component to this thesis. This section therefore aims to clarify the musical perspective

that has inspired me to perform this research and the context in which it can be understood. To do so, I explain my view on the artistic processes of the (electronic) music producer on the one hand (Section 1.2.1) and of the DJ (Section 1.2.2) on the other hand. These are two use cases that I am personally familiar with and that have been a fundamental motivation in the projects that I undertook throughout my research. I furthermore explain how I think that MIR technologies and tools can fit into these practices.

1.2.1 Making (electronic) music and the potential for MIR technology

Let us first consider the practice of music *making*. The following discussion elaborates on what I consider to be important aspects of the music making practice: finding inspiration, exploring ideas, and converging to a finished and polished piece of music.

Making music often starts from some source of *inspiration*. Some recurring sources of inspiration that I think are common among many musicians – including myself – are: a desire to express an emotion or a thought; an existing piece of music, e.g. with the intent to remix it or to sample parts from it to use in a new composition; samples, loops or synthesizer presets[3] that one comes across while exploring their music making resources; "happy accidents", i.e. interesting sounds, melodies, musical combinations or arrangements accidentally discovered throughout the music making process etc. This inspiration then leads to a musical idea or concept; concretely, this could be for example a particular rhythm or a melody, lyrics for a new song, an inspiring chord progression, a synthesizer patch, an audio effect applied to a certain sound that results in an

[3] Electronic musicians often have a collection of resources that they use in their compositions. This collection often includes samples (short recordings or synthesized sounds of individual notes played on a particular instrument, synthesizer or drum kit), loops (recordings of short melodies or rhythms containing one or more instruments), and software or hardware synthesizers with a corresponding collection of presets or "patches" for the synthesizer settings.

interesting soundscape, ...

This initial stroke of inspiration is then often followed by a phase of *exploration*. In this (often quite playful) process, the artist elaborates on the initial musical idea or tries to find further inspiration. For example, if the initial musical idea is an inspiring melody, then the artist could experiment with different instruments or synthesizers to play the melody with; they could try out different countermelodies to pair that melody with; they could perform a jam session with fellow musicians to build on the new idea etc. If the musical idea is a percussive rhythm, then developing that idea could involve actions such as searching for drum samples in the sample library that have desired sonic characteristics. If the musical idea is an existing song that the artist wants to remix, then they might cut up the original audio and rearrange the resulting audio slices; they might isolate certain instruments from the audio by applying audio filters and equalization and so on. This exploration often sparks new ideas to try out, and the result of this process is often a rough sketch of the music that the artist wants to make alongside an accumulation of ideas that may or may not fit in the final piece.

Finally, going from this rough sketch and collection of ideas to a polished piece of music requires a stage of *convergence*. This stage can roughly be divided into what I will call the "creative convergence" and the "technical polishing" of the music.

Creative convergence involves making artistic decisions about the accumulated collection of ideas: which ideas stay, which ones are discarded, and how ideas are combined and arranged into a coherent whole. This process can of course prompt the artist to identify "gaps" in the music, which in turn can trigger new cycles of inspiration, exploration and convergence, until the artist decides that the composition is finished.

There also is a strong technical aspect to finishing a music piece, including processes known as mixing and mastering. In order to understand these concepts, one could picture music as a layering of

many sound sources, which roughly correspond to the individual audio recordings of the different instruments in the music. If one were to simply play all sound sources simultaneously without any modification, then this would not sound great: certain instruments could be too loud while others are too quiet, or some instruments might clash in certain frequency ranges. The process of mixing involves performing various operations so that each sound source gets an appropriate "place" in the mix. These operations can include balancing the relative volume of each audio source; "panning" the sound sources to the left, to the right or to the center of the stereo image of the audio; applying equalization to each source to attenuate or boost certain frequencies; applying audio compression to control the dynamics of each source; etc. After mixing often comes a mastering step, where the aggregated (mixed) audio undergoes a final set of similar operations so that the final result sounds clear and as loud as intended. The result of all these processes is a finished and polished piece of music.

This discussion of course gives but a brief sketch of the creative process of an artist, but it illustrates the importance of having access to the right tools, effects and sources of inspiration in order to realize the aforementioned artistic goals. For example, if finding appropriate drum samples in their sample library is tedious and slow, then this might disrupt the artist's creative flow when exploring ideas. Simple audio filtering operations might not be precise enough to obtain a separation of instruments from an original recording that is clean enough to be used in a remix. Luckily, MIR technology can expand the artist's toolkit with powerful instruments to solve these issues. For example, fine-grained sound source separation algorithms allow to very precisely manipulate an existing song or a loop while creating a remix; machine learning-driven audio effects (such as the one presented in Chapter 6) allow to perform highly non-trivial musical transformations; symbolic music generation tools can support the artist in exploring ideas, e.g. by automatically generating a percussive backing while the artist

plays the melody, and in this way establishing a human-A.I. co-creative jamming session. This expands the creative palette for the artist and offers new ways to find inspiration, to explore existing ideas, and to express themselves in.

I furthermore see many opportunities for MIR to help to organize large collections of source material, such as sample and loop libraries or synthesizer patches. MIR could namely help to organize or view these collections based on musical properties that are automatically extracted from the audio of the items in the collection. This can help artists to explore their assets in more musically meaningful ways, see their collection from a new viewpoint and as such be inspired or see patterns that otherwise remained elusive, and it can reduce the time spent on routine tasks.

MIR tools can also be used to provide an informed starting point for certain parts of the composition, or to automate some sub-processes altogether. For example, a machine learning-driven mixing tool could allow the artist to continue exploring and combining musical ideas without having to interrupt their creative flow because they have to perform some standard mixing operations. By reducing the time spent on time-consuming routine tasks, these technologies could allow the artist to stay in their creative flow and to focus on what they find most important in their music. While such automated tools can be useful, the artist as well as the developer of these tools should be mindful of their limitations and biases when using or developing them. Chapter 7 discusses this limitation of automated MIR tools in more detail.

1.2.2 DJing and the potential for MIR technology

Secondly, let's consider the art of DJing, which has been an important source of inspiration for this thesis. The most common image of the DJ is that of the performer: the person at a party who is behind the mixing decks, who chooses the songs to play and as such tries to ensure that everyone has a good time. There is much more to the art of DJing besides this aspect of performance, though, as

will become clear from the following discussion.

In essence, the art of DJing involves creating a curated selection of music tracks and playing those tracks consecutively in such a way that the music transitions from one track to the other in some musically motivated way. The resulting performance is called a DJ mix or a DJ set. The goal of the performance is to create a certain atmosphere or to communicate particular emotions to the audience that fit the occasion for which the DJ set is made or performed[4]. To this end, a DJ often takes into account the emotion evoked by the individual songs in their selection, and plays them in such a way that an overarching progression of emotions is established. As such, the mix progresses through cycles of build-up, suspense, climax, surprise, cool-down, relaxation and so on, not unlike an individual song or a piece of classical music. A DJ set can therefore be seen as a new musical composition in itself, where the building blocks are not individual music notes and instruments but instead complete music tracks. Very often, a DJ also ensures that the transitions between songs are musical in some sense. For example, the DJ often aligns consecutive songs with the same metric grid (by using techniques known as *time-stretching*, *beatmatching* and *phrasing*; also see Chapter 5) or gradually introduces different musical elements of the next song while avoiding clashes with the music that is already playing (by using techniques such as e.g. volume fading and equalization). The DJ can also use the transition as a means of artistic expression: for example, a quick or even abrupt transition (a "cut") can create a moment of surprise, contrast, tension or climax, while a smooth and long blend of two songs could evoke feelings of harmony.

A DJ fulfills several roles in order to perform their art. Firstly,

[4] Note that the DJ does not have to be a performer on a stage: for example, a person who creates a pre-recorded hiphop or chill house mix to listen to while studying is also a DJ. In this case, the intended atmosphere might be one of concentration, focus and relaxation.

the DJ is a *music collector*. A DJ often spends a lot of time listening to new music, looking for songs to use in their sets. The result is that a DJ usually possesses a large and continuously expanding music collection. In order for that collection to be useful, DJs will often have some systematic way to store and arrange their music. As such, the DJ is also a *music organizer*. Music library organization can involve actions such as grouping songs by genre, sub-genre, year, or era of release (e.g. '80s, '90s); annotating songs by how they would fit in a DJ set, for example by assigning a rating on a scale from 1 to 5 of how "energetic" songs are; assigning labels to songs to easily retrieve them based on certain properties, for example labeling them as "contains vocals", "contains a saxophone", "four-to-the-floor rhythm" and so on.

The DJ is also a *DJ set composer/performer*. Creating a DJ set can be done in different ways. One way is to improvise it completely, i.e. select songs from the entire music collection "on the fly" without a premeditated plan or ordering. In this case, the set composition and performance are intertwined into one process. The other extreme is to fully prepared the set in advance, with a track selection and ordering (and often also a corresponding transitioning technique between the tracks) that is deliberated before the mix is recorded or performed. In between these extremes, another strategy is to first select a subset of songs from the entire collection to work with for that particular set. The songs are usually selected to ensure thematic or musical coherence throughout the set, and limiting the number of options a priori can help to increase focus during the performance. With this limited selection, the DJ can already have a rough plan for a progression of emotions throughout the mix, or have some songs in mind with which they would like to start the mix, which songs could be key climaxes etc. Instead of preparing the DJ set in its entirety, the DJ could also prepare a few shorter playlists, each consisting of a couple of songs that the DJ knows will work well together. This allows the DJ to fall back to known and practiced segments in the mix, while still having the flexibility to improvise and change plans throughout the set.

Regardless of the level of improvisation, preparation and organization, it is clear from this discussion that the music library is one of the most important assets – if not the most important – of the DJ. It is therefore essential that they can easily navigate through it, retrieve songs from it, and find interesting songs and song combinations in it. I believe that MIR can play an important supporting role in these tasks, as I explain below.

Finding harmonious, surprising or otherwise interesting combinations of songs is in my opinion one of the most exciting aspects of DJing. It is immensely gratifying to find a sequence of tracks that fit very well together, and the search for such combinations offers a way to endlessly keep exploring and experiencing the music that one already knows and loves. One way in which MIR systems can support the DJ in finding such interesting combinations, is by tagging or grouping songs according to automatically extracted descriptors of the music. For example, an automatic drum transcription (ADT) algorithm – a specific type of AMT algorithm designed specifically for transcribing the percussion in music – could first annotate each song with the predominant percussive rhythm or rhythms found in it. Then, the MIR system could group together tracks with similar rhythms by means of a clustering algorithm. This gives the DJ a new lens to view their own collection through, which can help them to identify interesting song combinations that were not immediately visible before, or to find good matches in the library very quickly. This use case is what inspired me to perform research on ADT systems, as will be described in Section 1.3.

MIR technology also allows to search for songs in a music collection based on information that is latent in the audio signal. For example, an AMT system could be a building block of a query-by-example system. In order to retrieve a song, the DJ can enter an approximate melody or rhythm that is then matched with songs in the library containing similar melodies and rhythms. Another example is a system that first uses a music sound source separation algorithm to extract a sample for each distinct drum sound in the

recording (the research presented in Part I describes such an algorithm), and then analyzes the musical properties of these samples. The DJ could then retrieve songs from their library based on the acoustic properties of, for example, the snare drum in each recording. This can be interesting if they are looking to blend songs with similar-sounding percussive instruments. Other use cases for MIR systems – some of which already exist or are even quite common in contemporary DJing software – are the automated annotation of tempo, beats and musical structure, of the music genre, of the musical key in which the piece is composed, of some measure of the "energy level" of the music etc. Note that such a music library management system is valuable not only to DJs, but also to e.g. radio makers, movie producers and many other types of creative professionals and music enthusiast. All the information that is extracted from the music audio offers additional handles and filters for the DJ to use in managing and exploring their library, which I think benefits the musical expressiveness and serendipity.

Finally, the DJ could also use MIR tools in the performance of the DJ set. Examples include using music sound source separation technology to extract stems from songs, which could result in cleaner-sounding mixes compared to traditional filtering and equalization operations; using highly accurate automatic annotations of structural boundaries in the music to quickly jump to relevant cue points (also see Chapter 5); applying advanced machine-learning based audio manipulations as sophisticated audio effects (e.g. see Chapter 6) etc. As in the case of the music producer, I think that expanding the toolkit of the DJ with these technologies opens interesting artistic avenues, which ultimately results in even more exciting music to enjoy.

1.3 Contributions of this thesis

This section summarizes the main research contributions of this thesis, and it briefly relates them to the musical perspective from Section 1.2 that motivated me to perform this research.

1.3.1 Automatic drum mixture decomposition using NMFD

In the first part of this thesis, I investigate how to address certain issues in the context of automatic drum transcription (ADT), i.e. the subfield of AMT that focuses on transcribing the percussion in the music signal. This research is motivated by the limitations mentioned in Section 1.1.3, namely the pre-defined nature and rigidity of transcription and decomposition systems, the lack of fine-grained decomposition systems for specific instruments (such as for drums), and the dependence of these systems on large training datasets. The musical inspiration for this research stems from my affinity with breakbeat-based music and drum'n'bass in particular. As explained in Section 1.2.2, accurate ADT and fine-grained sound source separation algorithms can lead to powerful tools to manipulate drum recordings, which is interesting for producing this type (and other types) of music. They are also valuable from a DJ's standpoint, as they may facilitate music retrieval from the music library or support them in their performance.

In my research, I investigate the "non-negative matrix factor deconvolution" (NMFD) algorithm in more detail. NMFD has been used in recent ADT research, and it has several properties that make it interesting to address the aforementioned challenges. Firstly, NMFD doesn't require training on a large dataset, as opposed to other contemporary ADT approaches using neural networks. It furthermore performs not only a transcription of the drum signal, but rather a *decomposition*, where it also extracts a spectral "template" of each transcribed drum sound. This joint transcription and decomposition could lead to a fine-grained source separation method for percussive audio. Finally, NMFD performs transcription and separation by exploiting the assumption that distinct musical elements (e.g. percussive sounds) repeat unaltered within a recording. Because this assumption is generic and valid for almost all percussive sounds, NMFD might be able to also tran-

scribe mixtures with uncommon instruments or rhythms. However, despite its appealing properties, recent work on NMFD remains quite limited and typically applies the algorithm without informed adaptations to the drum mixture decomposition problem. I therefore attempt to fill this gap by incorporating certain desired properties of drum signals into this framework. The results of the work performed in this context are disseminated in Part I of this thesis.

1.3.2 Manipulating drum'n'bass using MIR techniques

Section 1.1.3 described various examples of how MIR technologies are used to create novel ways to interact with music. In Part II, I describe my own contributions in this area, namely two systems that use MIR techniques to transform drum'n'bass music. Drum'n'bass is a genre of electronic dance music, characterized by a strong focus on fast and complex percussive rhythms on the one hand and on synthesized and modulated basslines on the other hand. As an amateur DJ and music producer, I know this genre particularly well, and this has allowed me to infuse domain knowledge about it into my work. This genre is also interesting and challenging from an MIR point of view, due to the fast tempo, extensive use of syncopated drum patterns (so-called "breakbeats"), and the subtle differences between different subgenres of drum'n'bass. The contributions presented in Part II have been a way to test the capabilities of MIR technology to perform certain highly non-trivial musical tasks in an automated fashion. The algorithms that were developed to this end could furthermore be used as tools to support the DJ or the music producer in their creative processes.

In Chapter 5, I describe the implementation of an automated DJ system for drum'n'bass music. This system mimics the entire workflow of a human DJ, starting from an unannotated collection of music in the target genre. To compose a DJ mix, it needs to carry out several steps. Firstly, the system needs to detect the tempo and understand the metrical and high-level compositional structure of

the music. Secondly, it needs to extract various descriptors from the music – for example the musical key, the overall timbre or "atmosphere" of the music, and information about whether vocals are present or what the predominant rhythm in the track is – to find compatible tracks[5] and create a playlist for the DJ to play. Finally, the tracks in the playlist are mixed together to create a seamless DJ mix. In order to extract the required musical knowledge from the audio, various MIR techniques need to be implemented. A second challenge is that a very precise annotation of the musical structure is required in order to generate DJ mixes of high quality. For example, when two music tracks are mixed together, it is essential that the beats in those tracks align perfectly. If the beats are misaligned only slightly, then this results in a severe deterioration of the perceived mix quality. The presented system realizes this by tailoring the implemented MIR methods to the target genre, which advocates in favor of genre-specific MIR systems if a high accuracy of annotation is desired.

In Chapter 6, I demonstrate a music style transfer system for drum'n'bass. As mentioned before, drum'n'bass encompasses various subgenres. A subgenre is often defined by the use of a specific type of sounds or instrumentation, by specific compositional techniques that are used, and/or by the overall atmosphere of the music (e.g. "relaxing" and "light" vs. "energetic" or "dark"). Some subgenres are tailored more towards energetic live DJ sets, whereas others are more relaxing. The work in Chapter 6 investigates the possibility to automatically transform the audio from one subgenre of drum'n'bass to another. This system applies an unpaired image-to-image translation technique – based on a Generative Adversarial

[5] The word "track" is a commonly used (popular) term within the electronic music community to refer to individual pieces of music. Note that the word "song" makes less sense in this context: electronic dance music is often instrumental music, and if there are vocals, then they often play a supporting role to the entire arrangement (as opposed to e.g. pop *songs*, where the vocals and lyrics are often the main focus of the music).

(neural) Network (GAN) model – to music spectrograms. Such a system could be used as an advanced audio effect in a live performance, or as a tool to quickly prototype new musical ideas.

Finally, Chapter 7 concludes this thesis. After a short overview of the main research contributions, several potential directions for further research are discussed, in the context of ADT – both regarding the NMFD algorithm as well as more generally – and in the context of using MIR for creative applications.

1.4 Publication list

This dissertation is the culmination of previous work that has been published in the following papers, in chronological order of publication:

- Vande Veire L., De Bie T. From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass. In *Journal on Audio, Speech, and Music Processing. 2018, 13*, 2018. DOI: `10.1186/s13636-018-0134-8`. URL: `https://doi.org/10.1186/s13636-018-0134-8`,

- Vande Veire L., De Bie T., Dambre J. A CycleGAN for style transfer between drum and bass subgenres. In *Proceedings of the Machine Learning for Music Discovery Workshop (ML4MD) at the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, USA, 2019,

- Vande Veire L., De Boom C., De Bie T. Adapted NMFD update procedure for removing double hits in drum mixture decompositions. In *13th International Workshop on Machine Learning and Music at ECML-PKDD 2020 (MML2020)*, Ghent, Belgium, 2020, pp. 10-14,

- Vande Veire L., De Boom C., De Bie T. Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum

Mixture Decomposition. In *Electronics. 2021; 10(3):284*, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10030284. URL: https://doi.org/10.3390/electronics10030284.

References

[1] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015. ISBN: 978-3-319-21944-8. DOI: 10.1007/978-3-319-21945-5.

[2] Peter Knees and Markus Schedl. *Music Similarity and Retrieval*. Springer, Berlin, Heidelberg, 2016. DOI: https://doi.org/10.1007/978-3-662-49722-7.

[3] Stanley S. Stevens, John Volkmann, and Edwin B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[4] Judith C. Brown. "Calculation of a constant-Q spectral transform". In: *The Journal of the Acoustical Society of America* 89.1 (1991), pp. 425–434.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. "Transfer learning for music classification and regression tasks". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017), Suzhou, China* (2017).

[7] Chih-wei Wu and Alexander Lerch. "From Labeled To Unlabeled Data – on the Data Challenge in Automatic Drum Transcription". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris, France* (2018).

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

[9] Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. "Singing voice separation with deep U-Net convolutional networks". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017), Suzhou, China* (2017), pp. 323–332.

[10] Deezer. *GitHub - deezer/spleeter: Deezer source separation library including pretrained models.* URL: `https://github.com/deezer/spleeter#projects-and-softwares-using-spleeter` (visited on 03/29/2021).

[11] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020). Deezer Research, p. 2154. DOI: `10.21105/joss.02154`. URL: `https://doi.org/10.21105/joss.02154`.

[12] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. "Open-Unmix - A Reference Implementation for Music Source Separation". In: *Journal of Open Source Software* (2019). DOI: `10.21105/joss.01667`. URL: `https://doi.org/10.21105/joss.01667`.

[13] VPRO. *The AI Song Contest – VPRO International.* 2020. URL: `https://www.vprobroadcast.com/titles/ai-songcontest.html` (visited on 03/26/2021).

[14] vrtNWS. *Artificiële intelligentie in Vlaanderen: 3 opvallende projecten.* 2018. URL: `https://www.vrt.be/vrtnws/nl/2018/09/19/in-beeld-artificiele-intelligentie-in-vlaanderen/` (visited on 03/26/2021).

[15] Focus On Belgium. *Musical collaboration between Stromae and artificial intelligence, a world first.* 2017. URL: `https://focusonbelgium.be/en/culture/musical-collaboration-between-stromae-and-artificial-intelligence-world-first` (visited on 03/26/2021).

[16] University of the Arts London (UAL). *Creative Computing Institute announces ground-breaking AI collaboration with Massive Attack*. 2019. URL: https://www.arts.ac.uk/about-ual/press-office/stories/creative-computing-institute-announces-ground-breaking-ai-collaboration-with-massive-attack (visited on 03/26/2021).

[17] AIVA Technologies SARL. *The AI composing emotional soundtrack music*. URL: https://www.aiva.ai/ (visited on 03/26/2021).

[18] Amper Music. *AI Music Composition Tools for Content Creators — Amper Music*. URL: https://www.ampermusic.com/ (visited on 03/26/2021).

[19] Christine Payne. "MuseNet". In: *OpenAI* (2019). URL: https://openai.com/blog/musenet/ (visited on 03/26/2021).

[20] Google Magenta. *Magenta Studio (v1.0)*. 2021. URL: https://magenta.tensorflow.org/studio (visited on 03/25/2021).

[21] Ke Chen, Cheng-i Wang, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. "Music SketchNet : Controllable Music Generation via Factorized Representations of Pitch and Rhythm". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020), Montréal, Canada* (2020).

[22] Jeff Ens and Philippe Pasquier. "Flexible Generation with the Multi-Track Music Machine". In: *Extended Abstracts for the Late-Breaking Demo Session of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020), Montréal, Canada* (2020).

[23] Hao Hao Tan and Dorien Herremans. "Music FaderNets: Controllable Music Generation Based On High-Level Features via Low-Level Feature Modelling". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020), Montréal, Canada* (2020).

[24] Sander Dieleman. *Generating music in the waveform domain.* 2020. URL: `https://benanne.github.io/2020/03/24/audio-generation.html` (visited on 03/26/2021).

[25] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. "GANSynth: Adversarial Neural Audio Synthesis". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA* (2019). URL: `https://openreview.net/forum?id=H1xQVn09FX`.

[26] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. "DDSP: Differentiable Digital Signal Processing". In: *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa, Ethiopia* (2020). URL: `https://openreview.net/pdf?id=B1x1ma4tDr`.

[27] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. "Jukebox: A Generative Model for Music". In: *arXiv preprint arXiv:2005.00341* (2020).

[28] Jake Drysdale, Maciek Tomczak, and Jason Hockman. "Adversarial Synthesis of Drum Sounds". In: *Proceedings of the International Conference on Digital Audio Effects (eDAFx-20), Vienna, Austria* (2020).

[29] Javier Nistal, Stephan Lattner, and Gaël Richard. "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using Generative Adversarial Networks". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020), Montréal, Canada* (2020).

[30] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *Proceedings of the 34th International Confer-*

ence on Machine Learning (ICML 2017), Sydney, Australia (2017).

[31] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. "A universal music translation network". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA* (2019).

[32] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows". In: *Applied Sciences* 10.1 (2020). ISSN: 2076-3417. DOI: 10.3390/app10010302. URL: https://www.mdpi.com/2076-3417/10/1/302.

[33] Matthew E P Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. "AutoMashUpper: Automatic creation of multi-song music mashups". In: *IEEE/ACM Transactions on Speech and Language Processing (TASLP)* 22.12 (2014), pp. 1726–1737. ISSN: 23299290. DOI: 10.1109/TASLP.2014.2347135.

[34] Hadrien Foroughmand Aarabi and Geoffroy Peeters. "Music retiler: Using NMF2D source separation for audio mosaicing". In: *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, Wrexham, UK* (2018), pp. 1–7.

[35] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller. "Let It Bee – Towards NMF-Inspired Audio Mosaicing". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015), Málaga, Spain* (2015), pp. 350–356.

Part I

Decomposing Percussive Mixtures using Non-negative Matrix Factor Deconvolution

2

Background: Decomposing Percussive Recordings using Non-Negative Matrix Factor Deconvolution

In many types of music, percussion plays an essential role to establish the rhythm and the groove of the music. Algorithms that can decompose the percussive signal into its constituent components would therefore be very useful, as they would enable many analytical and creative applications.

In this thesis, I investigate the use of the non-negative matrix factor deconvolution (NMFD) algorithm for this purpose. Given the spectrogram of a percussive music recording, NMFD discovers a dictionary of time-varying spectral templates and corresponding activation functions, representing its constituent sounds and their positions in the mix. I will refer to the MIR task that I am considering as *automatic drum mixture decomposition*, i.e. the joint estimation of both the sound sources as well as their onset locations in

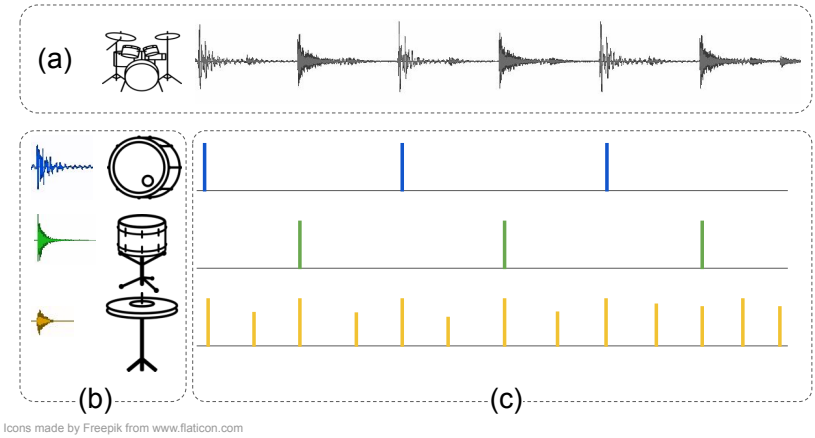


Figure 2.1: Conceptual illustration of the drum mixture decomposition task: a percussive mixture (**a**) is decomposed into prototypical samples of the used instruments (**b**) and the corresponding onsets (**c**).

a percussive mixture. Figure 2.1 shows a conceptual illustration of this task.

This introductory chapter provides the necessary background before presenting my own contributions to the NMFD algorithm in the next chapters. Section 2.1 gives a brief overview of the state-of-the-art in automatic drum transcription and decomposition. Comparing the properties of non-negative matrix factorization (NMF) based drum transcription algorithms with other contemporary approaches provides the motivation for using NMFD in my research. The NMFD algorithm is then introduced in Section 2.2.1, and its usage in previous works is discussed in Section 2.2.2. Finally, in Section 2.2.3, I explain the shortcomings that I observed when using the NMFD algorithm for analyzing percussive mixtures, and I introduce my own contributions, which are discussed in Chapters 3 and 4.

Large parts of this chapter, including Section 2.2, Section 2.2.1, Section 2.2.2, and Figure 2.1, are adapted versions of parts of text that were originally written for the following papers:

Vande Veire L., De Boom C., De Bie T. Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition. In Electronics. 2021; 10(3):284, 2021, ISSN: *2079-9292.* DOI: *10.3390/electronics10030284.* URL: *https://doi.org/10.3390/electronics10030284 [1] ,*

Vande Veire L., De Boom C., De Bie T. Adapted NMFD update procedure for removing double hits in drum mixture decompositions. In 13th International Workshop on Machine Learning and Music at ECML-PKDD 2020 (MML2020), Ghent, Belgium, 2020, pp. 10-14 [2] .

2.1 Drum Mixture Decomposition

The automatic drum mixture decomposition task aims to address the following question: given a musical mixture with multiple percussive instruments, can one jointly determine which instruments are present in the mixture and extract a prototypical "sample" of a single hit on each instrument, and at the same time transcribe the times at which each individual instrument is hit by the drummer (or is triggered by the drum programming software)? Such a decomposition enables many interesting applications. For example, a music producer could isolate a drum sample from an existing recording to use in their own compositions, or they could use such an algorithm for musicological analyses [3]. They could also replace a drum sound in an existing recording with a sample of their own, resynthesize the music by performing the inverse of the decomposition process, and as such obtain a *redrumming* of that recording without needing access to the original stems [4]. One could furthermore resynthesize each of the isolated sources into individual audio tracks and as such obtain stems for each percussive instrument, which again offers many creative possibilities. Alternatively, one could perform the inverse decomposition process for all sources but one, and effectively filter out one instrument from the original

recording. This can be useful for e.g. a DJ who wants to mix two drum loops, but leave out the kick drum from one of them.

Figure 2.1 shows a conceptual illustration of this MIR task. Note that in this context, the term "percussive instrument" refers to each percussive component in the recording with a distinct sound: for example, the kick drum, the snare drum, the closed hi-hat, the open hi-hat, a bongo, a cymbal hit etc. are all considered to be distinct percussive instruments (even though in other contexts, a drum kit could be considered as a single instrument, even though it consists out of multiple percussive "components").

The drum mixture decomposition problem described above is closely related to the problem of automatic drum transcription (ADT). ADT aims to detect and classify drum sounds events within a music recording, resulting in a list of onset locations for each transcribed instrument. Note that ADT only transcribes a drum recording and does not decompose it, and that there consequently is no inherent way to reconstruct (parts of) the audio recording from that transcription alone; hence, ADT does not enable many of the creative uses mentioned in the previous paragraph. Wu et al. [5] give a comprehensive overview of the state-of-the-art in ADT, and perform an in-depth comparison of these methods. They identify two classes of "activation-based" methods that currently dominate the state-of-the-art, namely, on the one hand neural network based systems using Recurrent Neural Network [6, 7] or Convolutional Neural Network [8] architectures, and on the other hand methods based on non-negative matrix factorization (NMF) [5, 9]. According to their analysis, neural network-based approaches outperform NMF-based methods in terms of transcription accuracy when a large and diverse training dataset with high-quality annotations is available. Such a dataset is not always available, however, and in the absence of such a dataset, NMF-based approaches offer a good alternative, as they do not require a data-hungry training procedure and still provide adequate performance if appropriately initialized. They are also more robust for drum mixtures with previously unseen instru-

ments [5]. Unsupervised transcription systems, such as the ones based on NMF, can furthermore be used to improve supervised approaches by leveraging them in semi-supervised learning schemes such as student–teacher learning [10].

In this work, we build upon the non-negative matrix factor deconvolution (NMFD) algorithm, an extension of NMF that explicitly models sounds with a temporal structure [11]. NMFD does not only discover the onset locations within the mixture: at the same time, it discovers a "template" of the constituent sounds that are responsible for each set of onsets. It is therefore enabled by design to address the drum mixture decomposition problem introduced at the beginning of this chapter. We furthermore use NMFD in an unsupervised fashion, i.e., after a best-effort initialization with templates of common percussive sounds, we allow the model to freely optimize the discovered templates to the target mixture without imposing any constraints on the sonic characteristics of the instruments that we expect to find. This is in contrast with most of existing ADT work, where often a predefined and fixed set of percussive instruments is considered. Therefore, we use the term automatic drum mixture *decomposition* in order to distinguish this use case from ADT, which is concerned with discovering the onset locations of an often predefined and fixed set of percussive instruments. In the next section, I describe the NMFD algorithm and present an overview of related work using NMFD for drum mixture transcription and decomposition.

2.2 Non-Negative Matrix Factor Deconvolution for Drum Mixture Decomposition

2.2.1 The Non-Negative Matrix Factor Deconvolution Algorithm

The non-negative matrix factor deconvolution (NMFD) algorithm [11] can be used for the drum mixture decomposition problem intro-

duced in Section 2.1. It decomposes a non-negative matrix $X \in \mathbb{R}_{\geq 0}^{N \times T}$ with N frequency bins and T time frames into a dictionary of K time-varying spectral *templates* $W^{(k)} \in \mathbb{R}_{\geq 0}^{N \times L}$, each L time frames long, and an *activation matrix* $H \in \mathbb{R}_{\geq 0}^{K \times T}$. The matrix X is modeled as the convolution of the templates with the activation matrix:

$$X_{n,t} \approx \hat{X}_{n,t} = \sum_{k=1}^{K} \sum_{\tau=1}^{L} W_{n,\tau}^{(k)} H_{k,t-\tau}, \tag{2.1}$$

where $H_{k,t-\tau}$ is zero when $t < \tau$. $W^{(k)}$ and H are updated iteratively using multiplicative updates in order to minimize a divergence measure $\mathcal{L}(X, \hat{X})$, typically the least squares loss, Kullback–Leibler (KL) divergence or Itakura–Saito divergence [12]. In this thesis, unless otherwise specified, I consider the KL-divergence objective function:

$$\mathcal{L}_{\mathrm{KL}}(X, \hat{X}) = \sum_{n,t} X_{n,t} \ln \left(\frac{X_{n,t}}{\hat{X}_{n,t}} \right) - X_{n,t} + \hat{X}_{n,t}. \tag{2.2}$$

which results in the following update rules for $W^{(k)}$ and H [13]:

$$\mathcal{L}_{KL}(X, \hat{X}) = \sum_{n,t} X_{n,t} \log \frac{X_{n,t}}{\hat{X}_{n,t}} - X_{n,t} + \hat{X}_{n,t}, \tag{2.3}$$

$$W_{n,\tau}^{(k)} \leftarrow W_{n,\tau}^{(k)} \frac{\sum_t H_{k,t-\tau} \left(X_{n,t} / \hat{X}_{n,t} \right)}{\sum_t H_{k,t-\tau}}, \tag{2.4}$$

$$H_{k,t} \leftarrow H_{k,t} \frac{\sum_\tau \sum_n W_{n,\tau}^{(k)} \left(X_{n,t+\tau} / \hat{X}_{n,t+\tau} \right)}{\sum_\tau \sum_n W_{n,\tau}^{(k)}}. \tag{2.5}$$

The templates $W^{(k)}$ can be interpreted as short spectrograms of length L that model the constituent sounds of the mixture. The corresponding activation curves H_k, i.e., the rows of H, describe where in the recording these sounds occur. In order for this interpretation to make sense, each sound should repeat itself almost unaltered throughout the recording, so that the templates captured

by $W^{(k)}$ can be "copied and pasted" at locations specified by H_k. This is a reasonable assumption for percussive instruments: hits on the same instrument will all sound approximately the same and will decay approximately equally fast, provided that the playing technique is consistent.

2.2.2 State-of-the-art in Drum Mixture Transcription and Decomposition using NMFD

NMFD has already been applied successfully for automated drum transcription and drum separation tasks [3, 12, 14–17]. For example, Laroche et al. [15] use a combination of NMF and NMFD in order to perform harmonic-percussive sound separation, modeling the non-percussive sounds using NMF and the percussive sounds using NMFD with predefined and fixed templates $W^{(k)}$. The percussive template dictionary is constructed by hand prior to decomposition, and the separated harmonic and percussive audio are obtained by means of Wiener filtering [18]. Lindsay-Smith et al. [14] investigate the use of sparsity constraints on the activations H in order to obtain impulse-like onsets. Ueda et al. [16] rewrite NMFD for drum transcription within a Bayesian framework, and impose a constraint on the time-quantized score S, which is derived from the activations H.

The aforementioned works apply NMFD to discover the activations H, for the purpose of ADT or audio source separation. In other works, NMFD has also been applied to capture the constituent drum samples in a recording as faithfully as possible in W; this is typically done in a score-informed setting, wherein the exact onsets of each instrument (and consequently H) are assumed to be known. NMFD is used as such in [17], where the extracted percussive sounds are subsequently used to validate a transient restoration technique that is applied when converting spectral representation back to an audible waveform. In [3], the authors apply NMFD to estimate the drum sounds in the Amen Break, a well-known drum

solo recording, in a score-informed setting. They observe that the unconstrained application of NMFD can lead to cross-talk artifacts, and they therefore propose two extensions to purify the extracted templates.

While the cited works illustrate the effectiveness of NMFD for drum mixture transcription and (score-informed) decomposition, they also share the shortcoming that NMFD is usually applied in a constrained setting. When NMFD is applied for ADT, i.e., to discover the activations H, then the templates W are usually predefined and kept fixed during optimization. This limits the application to drum mixtures where a reasonably accurate approximation of the constituent drum sounds is known in advance. On the other hand, when NMFD is applied to discover the constituent sounds, i.e., the templates W, then this is done in a score-informed setting, where H is assumed to be known in advance. With the work in [16] as an exception, we note the absence in the literature of a successful application of NMFD where both W and H are optimized jointly[1]. Such a joint optimization could be useful, though, as it would allow to decompose a drum mixture for which neither the exact onsets H nor a sufficient approximation of W are available in advance. In this thesis, I therefore investigate the application of NMFD to jointly decompose a drum mixture into its templates W and their activations H.

2.2.3 Contributions to the NMFD algorithm for automatic drum mixture decomposition

In this thesis, I address two issues that occur when NMFD is used for the decomposition of percussive spectrograms. Both are a result of the fact that the algorithm is too unconstrained and does not have any knowledge of what makes a decomposition informa-

[1] Note that there are examples in the literature of the application of (regular) NMF for automatic drum transcription with a joint optimization of the (one-dimensional) templates W and the activations H [9, 19].

tive and musically valid. The application of NMFD to decompose drum mixtures therefore often leads to undesired artifacts and solutions [3, 14, 20]. Consequently, additional measures are required to guide the optimization to the desired solution.

The first issue is that the time instants that are discovered by NMFD are often not "impulse-like", i.e. they do not localize the repetition of the templates at discrete time points but they are instead spread out over time. This does not make sense in the context of a drum mixture, and it also contradicts the interpretation that NMFD describes a mixture as the repetition of short templates through time. Instead, individual sounds in the mixture are often decomposed as the sum of several overlapping templates, making it hard to discern from the activations where in a mixture a certain drum hit occurs. In Chapter 3, I propose to address this issue by enforcing impulse-like behavior on the activations H. This effectively leads to sparse and impulse-like activation patterns, ensuring that the aforementioned interpretation of NMFD holds, and leading to more interpretable decompositions of drum mixtures.

The second issue that I address is that NMFD often captures *sequences* of drum strokes in long templates, instead of a single drum hit. Note that often, this issue cannot be resolved by simply choosing a shorter template length: the decomposition of some mixtures requires long templates because some drum hits in it have such a long decay. Instead, in Chapter 4, I propose an ad-hoc modification that detects the emergence of secondary onsets in the templates during optimization, and overwrites any excess drum hits in a template as soon as they start to appear. I illustrate that this indeed reduces the number of "double hits" in the discovered templates.

References

[1] Len Vande Veire, Cedric De Boom, and Tijl De Bie. "Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition". In: *Electronics* 10.3 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10030284. URL: https://www.mdpi.com/2079-9292/10/3/284.

[2] Len Vande Veire, Cedric De Boom, and Tijl De Bie. "Adapted NMFD update procedure for removing double hits in drum mixture decompositions". In: *13th International Workshop on Machine Learning and Music at ECML PKDD 2020 (MML2020), Ghent, Belgium* (2020), pp. 10–14.

[3] Christian Dittmar and Meinard Müller. "Reverse Engineering the Amen Break – Score-Informed Separation and Restoration Applied to Drum Recordings". In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 24.9 (2016), pp. 1531–1543. ISSN: 23299290. DOI: 10.1109/TASLP.2016.2567645.

[4] Patricio López-Serrano, Matthew E. P. Davies, Jason Hockman, Christian Dittmar, and Meinard Müller. "Break-Informed Audio Decomposition For Interactive Redrumming". In: *19th International Society for Music Information Retrieval Conference (ISMIR 2018) - Late-Breaking/Demos Session, Paris, France* (2018).

[5] Chih-wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, and Meinard Müller. "A Review of Automatic Drum Transcription". In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 26.9 (2018), pp. 1457–1483. DOI: 10.1109/TASLP.2018.2830113.

[6] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. "Drum Transcription via Joint Beat and Drum Modeling using Convolutional Recurrent Neural Networks". In: *Proceedings of the 18th International Society for Music Infor-*

mation Retrieval Conference (ISMIR 2017), Suzhou, China (2017), pp. 150–157.

[7] Carl Southall, Ryan Stables, and Jason Hockman. "Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016), New York City, USA* (2016).

[8] Carl Southall, Ryan Stables, and Jason Hockman. "Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017), Suzhou, China* (2017), pp. 606–612.

[9] Christian Dittmar and Daniel Gärtner. "Real-Time Transcription and Separation of Drum Recordings Based on NMF Decomposition." In: *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx 2014), Erlangen, Germany* (2014), pp. 187–194.

[10] Chih-wei Wu and Alexander Lerch. "From Labeled To Unlabeled Data – on the Data Challenge in Automatic Drum Transcription". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris, France* (2018).

[11] Paris Smaragdis. "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs". In: *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)* (2004), pp. 494–499. ISSN: 03029743.

[12] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrangey. "On Automatic Drum Transcription using Non-Negative Matrix Deconvolution and Itakura Saito Divergence". In: *Proceedings of the 2015 IEEE International Conference on Acous-*

tics, Speech and Signal Processing (ICASSP 2015), Brisbane, Australia (2015), pp. 414–418.

[13] Mikkel N. Schmidt and Morten Mørup. "Nonnegative matrix factor 2-D deconvolution for blind single channel source separation". In: *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2006), Charleston, SC, USA* (2006), pp. 700–707. ISSN: 03029743. DOI: 10.1007/11679363_87.

[14] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. "Drumkit Transcription via Convolutive NMF". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx 2012), York, UK* 1.3 (2012), pp. 15–18.

[15] Clement Laroche, Helene Papadopoulos, Matthieu Kowalski, and Gael Richard. "Drum extraction in single channel audio signals using multi-layer Non negative Matrix Factor Deconvolution". In: *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017), New Orleans, LA, USA* (2017), pp. 46–50. ISSN: 15206149. DOI: 10.1109/icassp.2017.7952115.

[16] Shun Ueda, Kentaro Shibata, Yusuke Wada, Ryo Nishikimi, Eita Nakamura, and Kazuyoshi Yoshii. "Bayesian Drum Transcription Based On Nonnegative Matrix Factor Decomposition With A Deep Score Prior". In: *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019), Brighton, UK* (2019), pp. 456–460.

[17] Christian Dittmar and Meinard Müller. "Towards transient restoration in score-informed audio decomposition". In: *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx 2015), Trondheim, Norway* (2015), pp. 1–8.

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

[19] Chih-Wei Wu and Alexander Lerch. "Drum Transcription Using Partially Fixed Non-Negative Matrix Factorization With Template Adaptation". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015), Málaga, Spain* (2015).

[20] Julian M. Becker and Christian Rohlfing. "Custom sized non-negative matrix factor deconvolution for sound source separation". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)* 3 (2014), pp. 2124–2128. ISSN: 15206149. DOI: 10.1109/ICASSP.2014.6853974.

3

Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition

The non-negative matrix factor deconvolution (NMFD) algorithm, introduced in Chapter 2, decomposes a spectrogram of a percussive recording into a dictionary of time-varying spectral templates and corresponding activation functions, representing the constituent sounds of the mixture and their positions in it. We observe, however, that the activation functions discovered by NMFD do not show the expected impulse-like behavior for percussive instruments. This is problematic, as this makes it hard to discern when the drum hits are detected in the recording.

In this chapter, I describe a modification of NMFD that enforces the expected impulse-like behavior for percussive instruments, by specifying that the activations should take binary

values: either an instrument is hit, or it is not. To this end, the activations are rewritten as the output of a sigmoidal function, multiplied with a per-component amplitude factor. Additionally, a regularization term is defined that biases the decomposition to solutions with saturated activations, leading to the desired binary behavior. Experiments on a publicly available dataset of percussive recordings shows that incentivizing the activations to be binary indeed leads to the desired impulse-like behavior, and that the resulting components are better separated, leading to more interpretable decompositions.

The research presented in this chapter has been published in the Special Issue on "Machine Learning Applied to Music/Audio Signal Processing" of the MDPI Electronics journal. The corresponding publication is:

Vande Veire L., De Boom C., De Bie T. Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition. In Electronics. 2021; 10(3):284, 2021, ISSN: *2079-9292.* DOI: *10.3390/electronics10030284.* URL: *https://doi.org/10.3390/electronics10030284 [1] .*

Note that this chapter has been modified with respect to the aforementioned publication, more specifically in Section 3.2.2 and Section 3.4.2. This is because after the paper had been published, an inconsistency was discovered in the code that affected the experimental results of the evaluation of the different optimization strategies for the proposed model. The results for the other experiments (e.g. the comparison with the sparse NMFD baselines) are unaffected and the main conclusions of this work remain the same as in the original publication.

3.1 Motivation for a Sigmoidal Model for the Activations

Percussive instruments are hit very briefly by some percussion mallet or beater; this implies that the discovered activations should be impulse-like, as the produced sounds result from excitations that are in fact impulses. This is not enforced by the original NMFD model, however; consequently, when applied to drum mixtures, we observe

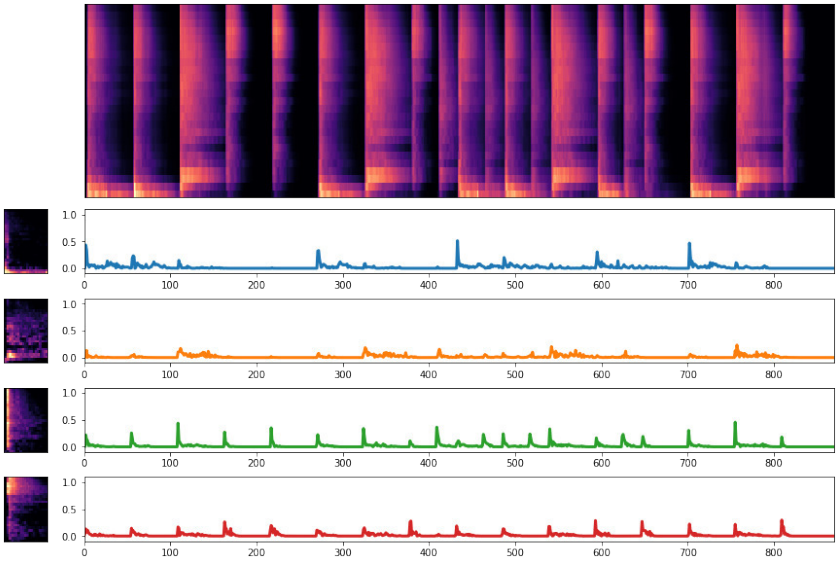


Figure 3.1: Decomposition of a percussive recording using non-negative matrix factor deconvolution (NMFD). The activations are not impulse-like, and contain noisy regions where it is difficult to detect individual drum hits.

that NMFD often does not lead to the expected impulse-like activations, as shown in Figure 3.1. This example illustrates that NMFD discovers activations with a sharp initial peak when a percussive hit occurs in the mixture, succeeded by a pseudo-exponentially decaying "tail' of small activation values. There are also small activations throughout each activation curve that do not succeed a larger peak, which makes the decomposition hard to interpret: do these activations correspond to a detected drum hit, or not?

To address these shortcomings, additional constraints are needed to guide the decomposition process. One approach would be to enforce an L1 sparsity constraint on the activations H [2, 3]. This encourages the algorithm to "move" as much information as possible from the activations H to the templates W, which would

lead to sparser and potentially more impulse-like activations. This approach has a drawback, however, i.e., it also penalizes correct activations. This biases the model to capture sequences of successive drum strokes within a single template, in order to keep the activations as sparse as possible [3, 4].

In this chapter, we present an alternative approach in order to achieve the desired impulse-like behavior: we enforce that the activations take binary values, i.e., either an instrument is hit, which yields an activation value of 1, or an instrument is not hit, which yields an activation value of 0. The relation between this constraint and the desired impulse-like behavior becomes clear when considering that enforcing such binary activations rules out the aforementioned "tails" and unclear activations: either an instrument is hit, or it is not, and values in-between 0 (not hit) and 1 (hit) are discouraged. An advantage of this constraint is that it does not penalize legitimate peaks, as opposed to a sparsity constraint.

To achieve the proposed binary activations, we redefine the activations in the model as the logistic function of a logit-activations matrix, and we impose a regularization term that pushes the activations towards the saturating regions of the logistic curve during optimization. As such, non-binary activation values are discouraged, and the activation values will be pushed to either 0 or 1 as much as possible. Of course, different sources can be present in the mix at different volumes: this is modeled by multiplying the binary activations with a per-component amplitude factor. A log-power spectrogram representation is used, as this further reduces the impact of velocity differences and emphasizes the binary behavior of the onsets. Note that we choose to maintain a continuous transition between the two saturated states, instead of choosing for a fully discrete quantization of the activation values: this ensures that the model and objective function remain differentiable so that the optimization procedure is tractable, and additionally allows some flexibility in activation values. Through evaluation on a public dataset, we show that these adaptations lead to decompositions with the desired impulse-like activations, and we illustrate by

means of an example that this can make the obtained unsupervised decompositions more interpretable.

3.1.1 Contributions

The main contributions presented in this chapter are the following.

- We reformulate the activations in the NMFD model as the product of a per-component amplitude factor, representing the relative volume of each component, with the time-varying activations for each component.
- These time-varying activations are defined as the output of a saturating sigmoidal function, and we propose a novel regularization term that combined with these saturating activations leads to binary activations. We show that in the context of automatic drum mixture decomposition, the activations are not only binary, but also become impulse-like as a consequence of this method.
- We propose different strategies and techniques to optimize the proposed model, and we rigorously evaluate their efficacy in minimizing the overall objective function for the decomposition.
- We propose metrics to evaluate the unsupervised decomposition of drum mixtures. With these, we show that the proposed algorithm achieves more impulse-like activations compared to unconstrained NMFD and sparse NMFD, making it better suited to the properties of percussive mixtures, while yielding a good decomposition and spectrogram reconstruction quality.

3.1.2 Structure of this Chapter

The remainder of this chapter is structured as follows. Section 3.2 introduces the modified NMFD algorithm and the procedure that

is used to optimize this model. Section 3.3 describes the baseline models, dataset, metrics, and experimental details. Section 3.4 discusses the experimental results, and Section 3.5 concludes the chapter and outlines directions for further research.

3.2 NMFD with Saturating Activations

3.2.1 Sigmoidal NMFD Model

The *logistic function* $\sigma(\cdot)$ is defined as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{3.1}$$

For large positive values of the input, $\sigma(x)$ saturates to 1, and for large negative values of the input, it saturates to 0. If $y = \sigma(x)$, then x is also called the *logit* of y.
We rewrite the NMFD model using saturating activations:

$$X_{n,t} \approx \hat{X}_{n,t} = \sum_{k=1}^{K} \sum_{\tau=1}^{L} W_{n,\tau}^{(k)} \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right), \tag{3.2}$$

with $X \in \mathbb{R}_{\geq 0}^{N \times T}$, $W^{(k)} \in \mathbb{R}_{\geq 0}^{N \times L}$, $G \in \mathbb{R}^{K \times T}$ and $a \in \mathbb{R}^{K}$. X and $W^{(k)}$ are scaled to maximum amplitude 1. By comparing Equations (2.1) and (3.2), we see that for the sigmoidal model, $H_{k,t} = \sigma(a_k)\sigma(G_{k,t})$. The sigmoidal activations $\sigma(G_{k,t})$ capture the onsets for each component k, while the amplitudes $\sigma(a_k)$ capture the relative volume of each component, as different components $W^{(k)}$ and $W^{(l)}$ can be present at different volumes in the mix. Note that the logit-activations $G_{k,t}$ and the logit-amplitudes a_k can take negative values.

3.2.2 Objective Function

The main objective of the decomposition is to minimize the divergence between the input spectrogram and the approximation. In

this work, we use the KL divergence:

$$\mathcal{L}_{\mathrm{KL}}(X, \hat{X}) = \sum_{n,t} X_{n,t} \ln \left(\frac{X_{n,t}}{\hat{X}_{n,t}} \right) - X_{n,t} + \hat{X}_{n,t}. \tag{3.3}$$

We furthermore want the activations to be binary in nature: for each t, the template $W^{(k)}$ should either be fully active, $\sigma\left(G_{k,t}\right) \approx 1$, or not active at all, $\sigma\left(G_{k,t}\right) \approx 0$[1]. In other words, $\sigma\left(G_{k,t}\right)$ must be in a saturating region of σ for all k and t. To achieve this, we define an additional regularization term $\mathcal{L}_G$:

$$\mathcal{L}_G(G) = \sum_{k,t} \exp\left\{ - \left(\frac{G_{k,t} - \mu_k}{2} \right)^2 \right\}, \tag{3.4}$$

$$\mu_k = \sigma^{-1}\Big(\alpha_k \max_t(\sigma(G_{k,t})) + (1 - \alpha_k) \min_t(\sigma(G_{k,t})) \Big). \tag{3.5}$$

Here, σ^{-1} is the inverse of the logistic function, and $\alpha_k = 0.1$. This regularization term encourages all logit activations $G_{k,t}$ of the k^{th} component to lie as far away as possible from the logit μ_k of the center activation value $\sigma(\mu_k)$, and $\mathcal{L}_G(G)$ is minimal when all activations saturate to either 0 or 1. Choosing $\alpha_k = 0.1$ leads to a stronger penalization of smaller unsaturated values and a less strong penalization of larger unsaturated values. The motivation for this is that we care mostly about eliminating small and unclear activations. If an activation value becomes sufficiently large, then it does not matter as much whether that activation value is e.g. 0.7, 0.9 or 1.0, as it is still clear from this activation value that the model detected a "hit" on the instrument. Note that this also allows some flexibility to describe hits at varying velocities.

The sigmoidal NMFD model, Equation (3.2), thus optimizes the following objective function:

$$\mathcal{L}_{\mathrm{tot}}(X, \hat{X}, G) = \mathcal{L}_{\mathrm{KL}}(X, \hat{X}) + \gamma \mathcal{L}_G(G). \tag{3.6}$$

[1] Note that the logistic function $\sigma(x)$ is never exactly equal to either 0 or 1 for real values of the logit x; this is only the case in the limit for $x \to -\infty$ and for $x \to +\infty$, respectively. Therefore, it would be more correct to say that the activations take *approximately* binary values.

The hyperparameter γ weighs the relative importance of the regularization term $\mathcal{L}_G$ with respect to the spectrogram reconstruction objective $\mathcal{L}_{\mathrm{KL}}$; in this work, we set $\gamma = 1$.

3.2.3 Optimization Procedure

Optimization Procedure Overview

Like the original NMFD algorithm [5], the model parameters $W^{(k)}_{n,\tau}$, $G_{k,t}$ and a_k are optimized in order to obtain a minimal loss $\mathcal{L}_{\mathrm{tot}}$ by means of an iterative optimization procedure. First, the parameters are initialized as explained in Section 3.3.2. Then, G, W and a are updated iteratively as follows:

1. Calculate $\hat{X}$ using the most recent estimates for W, G and a, as in Equation (3.2);
2. Update the logit-activations G, see Section 3.2.3, Equation (3.8);
3. Calculate $\hat{X}$ again with the new estimate for G;
4. Update the templates $W^{(k)}$, see Section 3.2.3, Equation (3.11);
5. Calculate $\hat{X}$ again with the new estimate for W;
6. Update the component-wise amplitudes a, see Section 3.2.3, Equation (3.12);
7. Repeat these steps until convergence.

Additive Gradient-Descent Update for G

$\mathcal{L}_{\mathrm{tot}}$ is minimized with respect to G using gradient-descent:

$$G_{k,t} \leftarrow G_{k,t} - \eta_G \frac{\partial \mathcal{L}_{\mathrm{tot}}(X, \hat{X}, G)}{\partial G_{k,t}} \tag{3.7}$$

$$= G_{k,t} - \eta_G \left(\frac{\partial \mathcal{L}_{\mathrm{KL}}}{\partial G_{k,t}} + \gamma \frac{\partial \mathcal{L}_G}{\partial G_{k,t}} \right). \tag{3.8}$$

The learning rate η_G is a hyperparameter of the optimization procedure. The partial derivatives in Equation (3.8) expand to (see Appendix A.1):

$$\frac{\partial \mathcal{L}_{\mathrm{KL}}}{\partial G_{k,t}} = \sum_{n,\tau} \left(1 - \frac{X_{n,t+\tau}}{\hat{X}_{n,t+\tau}}\right) \sigma(G_{k,t})\sigma(-G_{k,t}) W^{(k)}_{n,\tau} \sigma(a_k), \tag{3.9}$$

$$\frac{\partial \mathcal{L}_G}{\partial G_{k,t}} \approx -\left(G_{k,t} - \mu_k\right) \exp\left\{-\left(\frac{G_{k,t} - \mu_k}{2}\right)^2\right\}. \tag{3.10}$$

Note that we regard μ_k as a constant in the derivation of Equation (3.10). The expression for $\frac{\partial \mathcal{L}_G}{\partial G_{k,t}}$ is thus only approximately correct when $G_{k,t} = \max_{t'}(G_{k,t'})$ or $G_{k,t} = \min_{t'}(G_{k,t'})$. We do so in order to avoid instabilities in the updates for the ultimate values of the activations, see Appendix A.1 for details.

Multiplicative update for W

W is optimized using a multiplicative update rule, which ensures that W remains strictly positive. Its derivation from Equation (3.6) is analogous to that in Schmidt and Mørup [6], see Appendix A.2. This gives

$$W^{(k)}_{n,\tau} \leftarrow W^{(k)}_{n,\tau} \frac{\sum_t \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right) \left(\frac{X_{n,t}}{\hat{X}_{n,t}}\right)}{\sum_t \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right)}. \tag{3.11}$$

After each update, $W^{(k)}$ is scaled to maximum amplitude 1 for each k.

Additive Gradient-Descent Update for a

$\mathcal{L}_{\mathrm{tot}}$ is minimized with respect to a using gradient-descent:

$$a_k \leftarrow a_k - \eta_a \frac{\partial \mathcal{L}_{\mathrm{tot}}(X, \hat{X}, G)}{\partial a_k}, \tag{3.12}$$

where η_a is the learning rate and with $\frac{\partial \mathcal{L}_{\text{tot}}}{\partial a_k}$ given by (see Appendix A.3)

$$\frac{\partial \mathcal{L}_{\text{tot}}}{\partial a_k} = \sigma(-a_k) \sum_{n,t} \left[\left(1 - \frac{X_{n,t}}{\hat{X}_{n,t}} \right) \sum_{\tau} W_{n,\tau}^{(k)} \sigma(a_k) \sigma(G_{k,t-\tau}) \right]. \quad (3.13)$$

Optimization Strategies to Escape Local Minima

The model parameters are updated by iteratively applying Equations (3.8), (3.11), and (3.12). This is, however, a delicate task, as it is prone to converge to poor local minima due to the regularization term $\mathcal{L}_G$. In the update for G, this regularization namely pushes $G_{k,t}$ away from μ_k, see Equation (3.10). If the algorithm has not converged yet, then this prevents new peaks to grow or existing peaks to shrink, even if this would eventually lead to a better optimum. Imposing $\mathcal{L}_G$ too strongly or too early during optimization could therefore hinder convergence to a better local minimum.

We therefore propose and evaluate three optimization strategies that could help to find better local minima, as detailed below. In general, the optimization happens in different stages: an *unconstrained warm-up stage* , an *explore-and-converge stage*, and an ultimate *finalization stage*.

The goal of the **unconstrained warm-up stage** is to make the algorithm converge from its initialization (see Section 3.3.2) to a rough approximation of the spectrogram and a first estimation of the activation functions. During this stage, $\gamma = 0$, so that in this initial exploration, the activations are free to converge to the values that best approximate the spectrogram. We furthermore set $\eta_G = 0.5$, and η_a is set to 0.02. The warm-up stage is 30 iterations long. Empirically, this leads to good results in our experiments.

After the warm-up stage comes the **explore-and-converge** stage, wherein the proposed saturation regularization is applied and where an optimal solution is sought for the decomposition problem. In this work, we consider the following strategies to execute this stage:

- Optimization strategy 0: straightforward optimization.
  In this strategy, $\mathcal{L}_G$ is applied with $\gamma = 1.0$ at each iteration, and μ_k is calculated as in Equation (3.5) with $\alpha_k = 0.1$.

- Optimization strategy 1: staged application of $\mathcal{L}_G$.
  In this strategy, we periodically enable and disable $\mathcal{L}_G$ by alternating between "saturation sub-stages" and "fine-tuning sub-stages", which each last several iterations. During a *saturation sub-stage*, $\gamma = 1$, so that the activations are pushed towards saturation. During a *fine-tuning sub-stage*, $\gamma = 0$, so that the model has time to make peaks grow or shrink against the direction imposed by $\mathcal{L}_G$, in order to escape poor local minima.

- Optimization strategy 2: moving μ_k throughout optimization.
  In this strategy, we impose $\mathcal{L}_G$ at each iteration, i.e., $\gamma = 1.0$ for each update. In order to avoid squashing small peaks too early and additionally provide an incentive to escape local minima, we move around the "center point" μ_k of $\mathcal{L}_G$ (Equation (3.5)) by changing α_k in each iteration. More specifically, for each update of G and component k, we set α_k to a random value drawn from a uniform distribution over the interval $(0.05, 0.25)$. We hypothesize that setting α_k to a relatively low value helps to boost relatively small peaks, and that randomly sampling α_k could help to escape local optima.

- Optimization strategy 3: combine strategy 1 and strategy 2.
  This strategy combines the two aforementioned strategies: $\mathcal{L}_G$ is enabled and disabled alternatingly, and when it is applied, μ_k is moved around by sampling α_k from a uniform distribution over $(0.05, 0.25)$ for each update of G.

During the explore-and-converge stage, we set $\eta_G = 0.2$, as we find that this leads to a good convergence in our experiments. The learning rate η_a for the amplitudes a remains unchanged, i.e., $\eta_a = 0.02$. We perform 180 iterations in total during this stage. For strategy 1 and strategy 3, each sub-stage is 30 iterations long,

so that the explore-and-converge stage consists of three repetitions of alternating saturation and fine-tuning sub-stages.

The **finalization stage** concludes the optimization process. It consists of a final 30 iterations, in which we set $\gamma = 1$, $\eta_G = 0.1$, $\eta_a = 0.02$, and $\alpha_k = 0.1$. This allows the algorithm to converge to the final solution.

In our experiments, we found that normalizing the gradients of G and a in Equation (3.8) and Equation (3.12) to a maximum amplitude of 1 for each component is important to ensure that the activations in each activation curve G_k grow equally quickly. Otherwise, one component could grow much quicker than the others and start to dominate the decomposition, often resulting in a poor local optimum of $\mathcal{L}_{\text{tot}}$ where only one component is active (also see Section 3.4.2).

3.3 Experimental Set-Up

3.3.1 Baseline Models

We consider two baseline models to compare with. The first baseline model uses the original NMFD formulation from Equation (2.1) without additional constraints. The second baseline adds an L1 sparsity constraint with weighing factor λ to the objective function, Equation (3.3):

$$\mathcal{L}(X, \hat{X}, H) = \mathcal{L}_{\text{KL}}(X, \hat{X}) + \lambda \mathcal{L}_{\text{L1}}(H) \tag{3.14}$$

$$= \mathcal{L}_{\text{KL}}(X, \hat{X}) + \lambda \sum_{k,t} |H_{k,t}|. \tag{3.15}$$

In our experiments, we consider sparsity weights of $\lambda = 1.0$ (strong sparsity constraint), $\lambda = 0.1$ (medium sparsity constraint), and $\lambda = 0.01$ (weak sparsity constraint).

For both the unconstrained NMFD baseline and the L1-constrained baselines, we use the update rules from Schmidt and Mørup [6], as we observed numerical instabilities for the original

NMFD update rules from the work in Smaragdis [5] when $W^{(k)}$ contains columns that are much smaller in amplitude than other columns, i.e., when the sample captured by $W^{(k)}$ is silent at certain points. Similar observations were made in Lindsay-Smith, McDonald, and Sandler [3]. As for the sigmoidal model, 240 update iterations are performed. We furthermore evaluate two optimization strategies for the sparse baselines. In the first, L1 regularization is applied throughout the entire optimization, starting from the first iteration. In the second, the L1 regularization is disabled for the first 30 iterations, i.e., λ is set to 0. The reason for this second optimization strategy is that applying the L1 regularization too early might hinder proper convergence. This allows to evaluate whether the baselines would benefit from an "unconstrained warm-up stage" as used for the sigmoidal model.

For all baselines, the spectral templates $W^{(k)}$ are initialized in the same way as for the sigmoidal model, see Section 3.3.2, and are scaled to max amplitude 1 after each update as in the sigmoidal model. The activations H are initialized with random values drawn from a uniform distribution over $(0, 10^{-3})$.

3.3.2 Model Initialization

L is set to 50, which at a sample rate of 44,100 Hz and short-time Fourier transform (STFT) hop size of 256 corresponds to a template length of 290 ms. We set the number of components K to the number of percussive instruments in the mixture, which we assume is known in advance.

The templates $W^{(k)}$ are initialized using an averaged spectrogram template of drum hits of four common drum instruments: kick drum, snare drum, hi-hat, and crash cymbal. These average templates are created using a small dataset of individual drum hits [7], by averaging the aligned spectra of the single-hit samples of the desired instrument type. The first four components are initialized with a kick, hi-hat, snare, and crash template; if $K > 4$, then the excess components are initialized by alternating between the hi-hat

template and the snare drum template. Each $W^{(k)}$ is also rescaled to a maximum amplitude of 1.

For the sigmoidal model, the logit-activations $G_{k,t}$ are initialized with random values drawn from a uniform distribution over the interval $(-5, -4)$ so that $\sigma(G_{k,t}) \in (0.0067, 0.018)$. The logit-amplitudes a_k are initialized to 2, so that $\sigma(a_k) \approx 0.9$.

3.3.3 Dataset

The algorithm is evaluated on the ENST dataset [8], which contains annotated recordings of percussion-only pieces performed by three drummers on three different drum kits using a variety of beaters (sticks, rods, mallets, and brushes). We only use the wet mix of the "phrase" recordings, i.e., short drum sequences in various popular styles. There are 135 phrases in total, varying in tempo (labeled "slow", "medium", and "fast" in the dataset), and complexity ("simple", i.e., straight and without ornaments, and "complex", i.e., with fill-ins and ornaments). We modify the recordings slightly by cutting away the last hit of each recording. The motivation for this is that the last hit often "rings out", i.e., it has a long decay, which cannot be appropriately modeled in the NMFD paradigm, given the fixed and limited template length L[2].

[2]Note that, while hits on the same component as the last hit should have the same decay time, we do not observe any issues with modeling these hits throughout the mixture. We note two reasons for this. First, when an instrument with a long decay is hit, it is typically hit again before the first hit can fully "ring out", i.e., its decay is "cut short" by the second hit. Second, the low-volume "tail" of the decay is often masked by the sound of subsequent hits on other components. These two effects effectively reduce the template length L that is required to model the sounds in the mixture, except for the last hit where these effects do not apply. An alternative solution would be to increase L in order to fully capture these hits with a long decay within a single template. We choose not to do so, however, as this makes the optimization of the NMFD algorithm hard and prone to error, as discussed in [4].

3.3.4 Spectrogram Representation

For the experiments presented in this chapter, we use a custom Mel-frequency-scale log-power spectrogram representation. This spectral representation is designed to reduce computation time to obtain the decomposition, while maintaining a sufficiently fine resolution in order to distinguish all relevant sounds in the mixture.

The audio sample rate is 44,100 Hz. First, the STFT power spectrogram $X = |\mathrm{STFT}(y)|^2$ of the audio y is calculated using a frame length of 2048, a hop size of 256, and a Hann window over the frames. Then, the values of the STFT spectrogram are rescaled to the range $(0, 1)$, and they are summed along the frequency axis over N adjacent, non-overlapping frequency bands. The boundary frequencies of these bands are spaced according to a Mel-scale between 0 Hz and 11,025 Hz. Finally, a small value $\epsilon_{\text{pre-dB}}$ is added to the power spectrogram, which is then converted to a decibel scale and scaled to the range $(\epsilon_{\text{post-dB}}, 1)$.

Using the Mel scale makes the low to mid frequencies much more prominent in the resulting spectrogram as compared to a linear scale spectrogram, where a higher proportion of the bins would be allocated for higher frequencies. Using non-overlapping windows ensures that each STFT bin only contributes to one Mel-scale bin, so that the resulting spectrogram is less blurred. We find that these properties help to better distinguish between different instruments, especially those with a prominent presence in the low and mid frequencies (kick drum, snare drum, toms, bongos, etc.). A small number of bins N significantly reduces the time needed to decompose the spectrogram. We find that a limited number of bins is sufficient for a good decomposition and choose $N = 25$. Adding a small value $\epsilon_{\text{pre-dB}}$ before the decibel transformation masks low-valued noise, so that the resulting dB-scale spectrogram optimally uses its value range to differentiate between relevant power differences, making it clearer and easier to decompose. Scaling the spectrogram values to values in $(\epsilon_{\text{post-dB}}, 1)$ after transforming it to a dB scale is required for the sigmoidal model to be able to approx-

imate all spectrogram values, while a minimum value of $\epsilon_{\text{post-dB}}$ is used to avoid numerical instabilities in various computations. We set $\epsilon_{\text{pre-dB}} = 10^{-7}$ and $\epsilon_{\text{post-dB}} = 10^{-9}$.

3.3.5 Evaluation Metrics

In our evaluation, we wish to quantify the quality of unsupervised decompositions of drum mixtures, and compare these quantities for the outcome of the sigmoidal model and of the baseline models. Note that the unsupervised nature of the decomposition makes it more difficult to rely on ground-truth transcriptions of the music, as due to the unsupervised character there is no guarantee that the extracted components would match one-to-one with instruments in the music. Turning NMFD into a transcription algorithm would require incorporating some supervision mechanism that guides the components $W^{(k)}$ to the desired musical interpretation, which is beyond the scope of this work. Therefore, alternative evaluation metrics need to be used that quantify the quality or "usefulness" of such unsupervised decompositions.

We consider a decomposition to be of a good quality if

- the spectrogram is approximated well;
- all onsets in the mixture are detected, with as few false onset detections as possible;
- different components have different activation patterns: this means they contribute to the spectrogram at different times, which might indicate a more meaningful differentiation between components; and
- the activations are impulse-like.

The metrics to quantitatively assess these criteria are described in the following subsections. Some metrics are calculated over the activations $H_{k,t}$; when evaluated for the sigmoidal model, $H_{k,t}$ should be substituted by $\sigma(G_{k,t})$ in these metrics.

Spectrogram Reconstruction Quality

The spectrogram reconstruction quality is measured using the mean absolute error (MAE) between the target spectrogram and its reconstruction:

$$\mathrm{MAE}(X, \hat{X}) = \frac{1}{NT} \sum_{n,t} \left| X_{n,t} - \hat{X}_{n,t} \right|. \tag{3.16}$$

As all spectrogram values are scaled between $(\epsilon_{\text{post-dB}}, 1)$, MAE values of different spectrograms can be compared.

Overall Onset Coverage

We measure whether each onset in the drum mixture is accounted for by the decomposition, although without considering instrument information. First, peak picking is performed on each row of H. A value $H_{k,t}$ at an offset t is considered to be a peak if it satisfies three conditions [9]:

1. $H_{k,t} = \max(H_{k,t-\tau_{\max}:t+\tau_{\max}})$,
2. $H_{k,t} \geq \mathrm{mean}(H_{k,t-\tau_{\text{avg}}:t+\tau_{\text{avg}}}) + \theta_{\text{thr}} \max_t(H_{k,t})$,
3. $t - t_{\text{prev}} > \tau_{\text{wait}}$,

where t_{prev} is the offset of the last peak detected before t and where the hyperparameters are set as $\tau_{\max} = 5$ (corresponding to 29 ms), $\tau_{\text{avg}} = \tau_{\text{wait}} = 10$ (58 ms). We vary the value of the peak picking threshold θ_{thr} within the range of $(0.1, 0.9)$ in order to evaluate its influence on the metric proposed below.

The detected peaks are then shifted by the "template offset" $\tau_{\text{off}}^{(k)}$, which is calculated as the smallest value of τ for which the envelope of $W^{(k)}$, $w^{(k)}[\tau] = \sum_n W_{n,\tau}^{(k)}$, is larger than the average envelope value: $\tau_{\text{off}}^{(k)} = \min\left(\{\tau : w^{(k)}[\tau] \geq \frac{1}{L}\sum_\tau w^{(k)}[\tau]\}\right)$. This is necessary as the percussive hit modeled by $W^{(k)}$ might be shifted by some offset $\tau_{\text{off}}^{(k)}$ in the template.

These peaks are then compared with the ground-truth annotations. A peak in the decomposition is considered a true positive if there is a ground-truth onset of any instrument within the tolerance interval of 29 ms around that peak; otherwise, it is a false positive. Ground-truth annotations for which there is no peak detected within the tolerance interval around it are false negatives. The precision, recall, and F-measure are calculated using these true positive, false positive, and false negative counts.

Note that this metric allows a ground-truth onset to be "covered" by multiple activation peaks and vice versa, and that peaks from any component can match with onsets from any instrument. We do not attempt to match components with specific instruments in the ground-truth annotations, as this is a difficult task that is prone to error and ambiguity, and we consider this beyond the scope of the unsupervised decomposition that is considered in this chapter.

Activation Curve Similarity

This metric quantifies how different the activations from each component are from the activations of any other component in the decomposition. We consider a decomposition to be of higher quality if the different activation curves are disentangled, i.e., they activate often at distinct times in the mixture. Each component then models drum hits that are not modeled by other components. On the other hand, a high similarity between activation curves indicates that multiple components often contribute to the same onsets, so that it could be difficult to figure out the relationship between instruments in the mixture and components in the decomposition.

Note that we expect the activations in the decomposition to have at least a low amount of similarity, as the onsets in different rhythmic instruments are often correlated and will coincide at least sometimes. However, an exceedingly high similarity value would be unexpected, as we expect distinct instruments to have at least some degree of uniqueness to their activations, and it is this undesired

behavior that we wish to detect by using this metric.

To quantify activation curve similarity for a given activation matrix H, each activation curve is first smoothed and made non-zero using a running mean operation:

$$\bar{H}_{k,t} = \frac{1}{2M+1} \sum_{u=-M}^{M} H_{k,t+u} + \epsilon_H. \tag{3.17}$$

We set M to 5, corresponding to a symmetric tolerance interval of 29 ms around each t. This smoothing allows to better compare two activation curves that capture the same onsets but that are slightly shifted with respect to each other. Then, the cosine similarity is calculated between every pair of rows in H. The small value $\epsilon_H = 10^{-52}$ ensures that comparing one row of H with an all-zero row in H still results in a meaningful metric value, for example, comparing two all-zero rows in H should result in a similarity value of 1. After calculating the pairwise similarity of all rows in H, we consider the minimum, mean and maximum similarity between any pair of rows to quantify the amount of differentiation between the activations for each decomposition.

A high value for the maximum similarity indicates that there are at least some components that detect more or less the same hits in the mixture, which is undesirable.

Peakedness Measure

This metric quantifies to what extent a decomposition is impulse-like, by comparing the original activation curve with a processed version in which peaks are accentuated and small values are removed. We define the *half wave rectification* operation $\mathrm{HWR}(x)[t]$ as

$$\mathrm{HWR}(x)[t] = \max(x[t] - \bar{x}[t], 0), \tag{3.18}$$

in which $\bar{x}$ is the smoothed version of x, see Equation (3.17). We furthermore define the *compansion* (compression-expansion) oper-

ation $\text{comp}_\kappa(x)[t]$ with exponent κ as

$$\text{comp}_\kappa(x)[t] = \max_u(x[u]) \left(\frac{x[t]}{\max_u(x[u])} \right)^\kappa. \tag{3.19}$$

If $\kappa > 1$, then $\text{comp}_\kappa(x)$ makes relatively small values even smaller compared to the maximum value of x, accentuating large values. If $\kappa < 1$, then $\text{comp}_\kappa(x)$ makes relatively small values larger. We then calculate the peak-accentuated version of H_k as

$$H_{k,t}^{\text{peaks}} = \text{comp}_{\kappa^{-1}} \left(\text{HWR}(\text{comp}_\kappa(H_k)) \right)[t], \tag{3.20}$$

with $\kappa = 3$. This operation should be understood as follows. First, the inner compansion accentuates the highest peaks in H_k, while making smaller peaks even smaller. The HWR operation then removes values that are smaller than the running mean around it, which further accentuates peaks and removes low-valued noise. The outer compansion then restores the peaks to their original relative scale, as long as they were not removed by the HWR operation.

The peakedness of an activation curve $H_{k,t}$ is defined as the ratio $\sum_t H_{k,t}^{(\text{peaks})} / \sum_t H_{k,t}$. If the ratio of the sum of values of $H_{k,t}^{(\text{peaks})}$ and $H_{k,t}$ is close to 1, then $H_{k,t}$ changed very little by the impulse-accentuating operation, i.e., it was already quite impulse-like itself. If the ratio is lower, however, then around the peaks in $H_{k,t}$ there must be low values that are removed by the HWR operation when calculating $H_{k,t}^{(\text{peaks})}$, meaning that the activations are less impulse-like.

We report the average of the peakedness values of all activation curves of H.

3.3.6 Implementation Details

Our NMFD implementation is loosely based on the NMFD implementation from López-Serrano et al. [10]. All code is made available on a public online repository [11].

3.4 Results

This section presents the evaluation of our method. Sections 3.4.1 and 3.4.2 present the evaluation on the ENST dataset. Section 3.4.3 presents a case study to visually illustrate the effects of our method.

The purpose of our evaluation is twofold. The first objective is to compare the performance of the sigmoidal model and of the baselines in terms of the proposed evaluation metrics. This comparison is presented in Section 3.4.1 and Table 3.1. In this evaluation, the sigmoidal NMFD model is optimized with a simplified and straightforward optimization strategy, i.e., optimization strategy 0 with a constant learning rate η_G. Comparing with this simplified algorithm helps us understand to what extent the observed improvements are caused by the proposed model itself, rather than by certain elements in the optimization strategy (also see Section 3.4.2). For completeness, Table 3.1 also reports the results for the sigmoidal model optimized with an alternative optimization strategy, i.e. strategy 2 with γ set to 0.1 during the explore-and-converge stage.

The second objective of the evaluation is to provide an in-depth analysis of the additional gains that can be achieved by using more advanced optimization strategies. This analysis is provided in Section 3.4.2. We furthermore perform an ablation study to quantify the impact of several techniques we use in the optimization of our model. From this evaluation, we conclude that some techniques help to achieve better local minima of the objective function $\mathcal{L}_{\text{tot}}$ more consistently, whereas other techniques are not as effective.

3.4.1 Evaluation on the ENST dataset

Table 3.1 shows the results of the evaluation on the ENST dataset.

As discussed in Section 3.3.1, the baselines are evaluated once with and once without an "unconstrained warm-up stage". We found that performing a warm-up stage for optimizing the sparse baselines leads to virtually the same results as not using that tech-

Table 3.1: Comparison of the performance of the NMFD baseline, the sparse NMFD baselines, and the proposed sigmoidal NMFD model on the evaluation metrics. For each metric, the mean value over all 135 phrases is shown (standard deviation between parentheses). The results for the peakedness metric for the strong sparsity baseline (*) are computed after discarding any all-zero activation curves in H. For each metric, the most optimal value is shown in bold in each column.

Algorithm	**MAE**	**Overall onset coverage**				**Activations similarity**			**Peakedness**
		Pr.	Rec.	F-score	F-score ($\theta_{\text{thr}} = 0.5$)	Min	Mean	Max	Mean
Unconstrained NMFD	**0.025** (**0.003**)	0.76 (0.15)	**0.94** (**0.09**)	0.83 (0.12)	0.63 (0.17)	0.45 (0.12)	0.63 (0.10)	0.79 (0.12)	0.42 (0.02)
NMFD + L1 sparsity ($\lambda = 0.01$)	0.026 (0.003)	0.77 (0.15)	**0.94** (**0.09**)	0.84 (0.12)	0.61 (0.18)	0.39 (0.13)	0.59 (0.10)	0.77 (0.12)	0.43 (0.02)
NMFD + L1 sparsity ($\lambda = 0.1$)	0.037 (0.005)	0.76 (0.16)	0.91 (0.09)	0.82 (0.12)	0.50 (0.18)	**0.04** (**0.10**)	**0.19** (**0.10**)	0.65 (0.21)	0.45 (0.08)
NMFD + L1 sparsity ($\lambda = 1.0$)	0.071 (0.009)	**0.88** (**0.15**)	0.84 (0.13)	**0.85** (**0.13**)	0.35 (0.17)	0.15 (0.31)	0.57 (0.25)	0.90 (0.26)	0.46* (0.13)
Sigmoidal NMFD (strategy 0, $\gamma = 1.0$, constant η_G)	0.044 (0.006)	0.83 (0.17)	0.78 (0.13)	0.79 (0.13)	0.70 (0.15)	0.07 (0.12)	0.23 (0.10)	**0.51** (**0.14**)	**0.72** (**0.06**)
Sigmoidal NMFD (strategy 2, $\gamma = 0.1$)	0.035 (0.004)	0.79 (0.19)	0.88 (0.11)	0.82 (0.14)	**0.73** (**0.13**)	0.12 (0.11)	0.28 (0.10)	0.56 (0.15)	0.67 (0.06)

nique, i.e., the outcome in terms of the metrics reported in Table 3.1 is exactly or almost exactly the same. For the sake of conciseness and not cluttering the Table, we therefore omit the results for the sparse baselines with a warm-up stage from Table 3.1. We conclude that the sparse baselines are not hindered in their convergence by applying regularization from the beginning of the optimization. The conclusions drawn in the following evaluation are therefore valid for all sparse baselines, regardless of whether or not an unconstrained warm-up stage has been applied.

Regarding **the spectrogram reconstruction quality**, the average MAE is low for most models, i.e., all spectrograms are approximated well. For the high sparsity baseline, the mean MAE is approximately twice as high as for the other models, suggesting that the L1 regularization in this baseline is too strong and leads to a worse spectrogram approximation. On average, the approximations by the medium sparsity baseline are comparable to those by the sigmoidal model in terms of MAE, and slightly worse than the unconstrained NMFD algorithm. This result is expected, as the unconstrained model optimizes only the reconstruction loss $\mathcal{L}_{\mathrm{KL}}$, while the other models have to take additional constraints into account.

In terms of **onset coverage**, all algorithms perform similarly in terms of F-measure, with slight differences in precision and recall. The baseline NMFD model and the weak sparsity model give a better recall, while the sigmoidal model and the high sparsity baseline lead to an improved precision. The sigmoidal model and high sparsity baseline thus yield fewer false positives at the expense of missing more ground-truth hits[3]. Based on the low-threshold on-

[3]The precision is equal to the ratio of the number of peaks in the activations that "match" a ground-truth hit over the total number of detected peaks. Therefore, an improved precision means that a higher proportion of peaks in the activations correspond with a ground-truth hit. The recall is equal to the ratio of the number of ground-truth hits that were detected, i.e., that have a "match" in the activations, over the total number of ground-truth hits. For the sigmoidal model and high sparsity baseline, the recall is lower than for the other baselines, but the precision is higher: hence, on average, fewer ground-truth hits are detected, i.e., a lower recall, but the peaks that are detected in the activations

set coverage metrics, all algorithms seem to perform approximately equally well in detecting the onsets in the mixture.

This conclusion changes when a high threshold is used for peak picking. Table 3.1 shows the F-measure when the peak-picking threshold is changed to $\theta_{\text{thr}} = 0.5$, i.e., in each activation curve, a peak is only considered if it is at least half as high as the largest value in the curve. In this case, the F-measure drops for all models; however, the decrease is much more severe for the baseline models, whereas the performance of the sigmoidal model remains relatively stable. The decrease in performance is most pronounced for the strong sparsity baseline. In other words, the activations discovered by sigmoidal NMFD are the least sensitive to the specific choice of peak picking threshold, which is an indirect indication that the activations are approximately equally high, i.e., they exhibit binary behavior. In the other baselines, there is more variation in peak height within each activation curve, and this increases with increasing L1 sparsity. For completeness, Figure 3.2 shows the evolution of the F-measure as a function of the peak picking threshold θ_{thr}. This again shows that the performance decrease for increasing θ_{thr} is more severe for the baselines with a stronger sparsity term, whereas the sigmoidal model maintains a much more stable performance for increasing θ_{thr}.

In terms of **activation curve similarity**, both the unconstrained NMFD baseline and the low sparsity baseline ($\lambda = 0.01$) have an average minimum and mean similarity that is considerably higher than of the other models. A non-zero minimum similarity is not necessarily undesired: percussive events of different instruments in the same recording are often correlated, so some similarity is to be expected. However, too much similarity might indicate the undesired result that the discovered components all represent parts of the same percussive onsets, leading to an entangled decomposition that is hard to interpret. Visual inspection of the decompositions (see Section 3.4.3) will indeed show that this is the case for the

---

are more likely to correspond with a ground-truth hit, i.e., a higher precision.

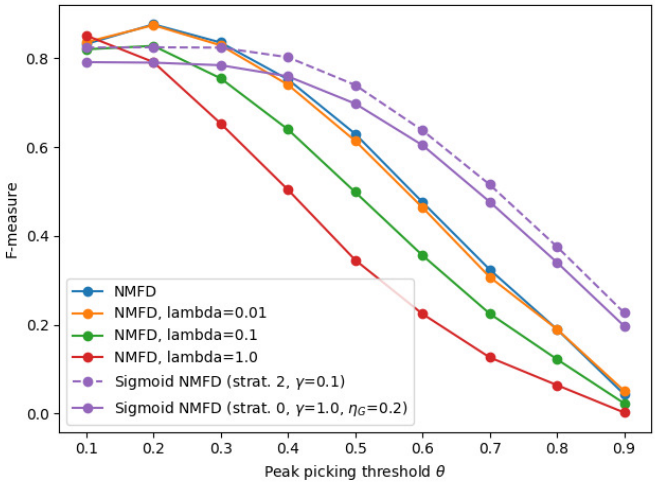


Figure 3.2: Onset coverage F-measure as a function of the peak picking threshold θ_{thr}.

unconstrained and low sparsity baselines.

When the L1 sparsity is too high, we observe a substantial increase in mean and maximum activation similarity. This is because in this case many activation curves are effectively "disabled" by becoming monotonically zero, so that only a fraction of the allocated number of activation curves is effectively used to capture onsets. All the "disabled" activation curves of course show a high similarity between each other.

The best performing baseline is the medium sparsity baseline ($\lambda = 0.1$). This baseline has a slightly lower average minimum and mean activation curve similarity than the sigmoidal model. However, as mentioned in Section 3.3.5, a low value of the similarity metric is to be expected, and it is hard to compare values of the metric when both comparands are reasonably low; therefore, we do not draw any conclusions from this observation. Nevertheless, as with the other baselines, this baseline also shows a considerably higher average maximum activation similarity compared to the sig-

moidal model, indicating that it creates decompositions that often contain at least some components that are highly correlated.

We conclude that in terms of activation curve similarity, the proposed sigmoidal model outperforms all baselines in terms of average maximum activation curve similarity, indicating that it on average makes better use of the allocated "capacity", i.e., the number of components K, to model distinct sound events in the mixture. This means that the decompositions from the sigmoidal model are more disentangled and therefore more likely to be interpretable.

Finally, the results for the **peakedness** metric show that the proposed approach indeed yields much more peaked activations than the non-regularized NMFD and sparse NMFD. A perhaps surprising result is that enforcing L1 sparsity does not lead to a notable increase in peakedness.

We conclude that the proposed sigmoidal model yields decompositions where the activations are considerably more impulse-like than the considered baselines, which is the desired outcome of the proposed approach. By design, the activation peaks are furthermore more uniform in height, which makes performing peak picking on the obtained activations less sensitive to the specific choice of the peak picking threshold. From the activation curve similarity, we conclude that the components are better disentangled, while the MAE and onset coverage metrics show that the algorithm maintains a good spectrogram reconstruction and onset detection quality. Both of these conclusions hold when the model is optimized with the simplified optimization strategy, as well as when a more advanced optimization strategy is used, suggesting that the improvements are not caused by the particular optimization strategy but rather by the model itself.

In Section 3.4.3, we show by example that these results can improve the interpretability of the decomposition.

3.4.2 Evaluation of the Optimization Strategies and Techniques

In this section, we evaluate the efficacy of the optimization strategies proposed in Section 3.2.3. The goal of this analysis is to evaluate to what extent more advanced optimization schemes lead to a better minimization of the loss $\mathcal{L}_{\text{tot}}$, compared to a more straightforward optimization strategy. More specifically, we consider the following settings:

- strategy 0, i.e., straightforward optimization with $\gamma = 1.0$ and "static" μ_k;
- strategy 1, i.e., a staged application of $\mathcal{L}_G$;
- strategy 2, i.e., setting μ_k to a random and relatively small value for each update of G;
- strategy 3, i.e., the combination of strategy 1 and strategy 2;
- each of the above, but with $\gamma = 0.1$ during the explore-and-converge stage, in order to evaluate the effect of applying the regularization less strongly during the exploration stage (for strategies 1 and 3, γ remains 0 during the fine-tuning substages). Note that in the following discussion, these strategies will still be evaluated by how well they minimize the original formulation of $\mathcal{L}_G$, i.e., with $\gamma = 1.0$.

We furthermore perform ablation experiments in order to assess the importance of

- the component-wise normalization of the gradients of G and a when performing the updates;
- the unconstrained warm-up stage, i.e., performing a few iterations of unconstrained optimization before $\mathcal{L}_G$ is applied; and
- the step-wise adaptation of the learning rate η_G throughout the optimization procedure.

We evaluate each strategy by their ability to minimize the objective function $\mathcal{L}_{\text{tot}}$. To do so, we calculate the *loss per timestep* $\mathcal{L}_{\text{tot}}/T$ for each decomposed spectrogram, and then report the average loss per timestep over all 135 examples in the ENST dataset. Dividing the loss of each decomposed spectrogram by the number of timeframes T of that spectrogram results in a quantification of the decomposition loss that is insensitive to the duration of the decomposed drum recording, so that we can appropriately average over all examples in the dataset, as the dataset contains recordings of varying lengths (a recording that is twice as long as another, but that is decomposed equally well, is expected to have a loss $\mathcal{L}_{\text{tot}}$ that is twice as high as the decomposition of the shorter recording, as $\mathcal{L}_{tot}$ scales linearly with the length of the decomposed mixture if a constant decomposition quality is assumed.). Additionally, the loss per timestep is reported for the two terms of the objective function, namely for the spectrogram reconstruction objective $\mathcal{L}_{\text{KL}}$ and for the regularization term $\mathcal{L}_G$. We also report on the different metrics defined in Section 3.3.5.

The results of this evaluation are shown in Table 3.2. Inspecting the results for the different optimization strategies leads to several observations. In terms of the metrics from Section 3.3.5, all strategies seem to perform comparably well, and better than the baseline models, see Table 3.1. On average, strategy 1 leads to a slightly higher loss per timestep than strategy 0 (and similar conclusions hold for strategy 3 vs. strategy 2, see below). The results for the individual loss terms show that this is because strategy 1 leads to a slightly better spectrogram approximation, i.e. $\mathcal{L}_{\text{KL}}$ is lower, at the cost of a larger penalty caused by the regularization term, i.e. $\mathcal{L}_G$ is higher. This is also reflected in the evaluation metrics: the experiments using strategy 1 lead to a lower MAE, indicating a better spectrogram approximation, at the expense of a lower F-score with a high peak picking threshold and a lower peakedness, both indicating a slightly less impulse-like result. Setting γ to 0.1 during the explore-and-converge stage leads to similar observations, although the differences in average loss per timestep are more pro-

nounced: applying the regularization term less strongly during this stage results in a better spectrogram approximation at the expense of the peakedness of the solution. On average, the strategies with $\gamma = 1.0$ often lead to better separated components as indicated by the mean and maximum activation similarity, which might be a desirable property of the decomposition.

Another observation is that strategy 2 seems to be ineffective. The overall loss per timestep $\mathcal{L}_{\text{tot}}/T$ as well as the individual loss terms $\mathcal{L}_{\text{KL}}/T$ and $\mathcal{L}_G/T$ are approximately the same in the experiments where strategy 2 is used compared to their counterparts where it is not used (strategy 0 vs. strategy 2, strategy 1 vs. strategy 3). The differences in terms of the evaluation metrics are minimal.

We repeat the experiment for strategy 2 with $\gamma = 0.1$, but without the component-wise normalization of the gradients of G and a when performing the updates. This leads to a considerably higher mean and standard deviation for the loss per timestep, indicating that normalizing the gradients component-wise indeed makes the algorithm's performance more consistent and reliable.

We furthermore perform an ablation study in order to assess the importance of the unconstrained warm-up stage at the beginning of the optimization. To do so, we repeat our experiments, but wherein $\mathcal{L}_G$ is enforced during the first 30 iterations, i.e., $\gamma = 0.1$ or $\gamma = 1.0$ (depending on the particular experiment) instead of $\gamma = 0$. The results are reported in Table 3.2 for strategy 2 with $\gamma = 0.1$ as well as for strategy 0; the results of the other strategies are omitted in Table 3.2 for conciseness, but are described in the following discussion.

For the strategies with $\gamma = 1.0$, not performing an unconstrained warm-up leads to considerably worse results. For strategies 0 and 2, there is a severe increase in the mean loss per timestep (from 0.25 to 2.89 for strategy 0, from 0.25 to 1.92 for strategy 2). For strategies 1 and 3, there is also a considerable increase in mean loss per timestep, but it is not as severe as for the other two strategies (from 0.27 to 0.51 for strategy 1, from 0.27 to 0.31 for

strategy 3); periodically disabling the regularization term during the explore-and-converge stage, a technique that is used in both strategy 1 and strategy 3, seems to help to recover from the poor initial convergence due to applying $\mathcal{L}_G$ too early in the optimization process.

For the strategies with $\gamma = 0.1$, the mean loss per timestep also increases, although not as drastically as with $\gamma = 1.0$. More specifically, in this case, not performing an initial convergence stage leads to a mean loss per timestep of 0.42, 0.46, 0.35, and 0.41 for strategies 0, 1, 2, and 3 respectively, compared to a mean loss per timestep of 0.35, 0.41, 0.34, and 0.41 originally. We suspect that setting γ relatively low at the beginning of the optimization and during the explore-and-converge stage allows the algorithm to still converge to a reasonable approximation of the spectrogram before $\mathcal{L}_G$ is applied with $\gamma = 1.0$ in the finalization stage, which leads to better results compared to setting $\gamma = 1.0$ throughout the entire optimization process.

We conclude that an unconstrained warm-up stage is essential for a proper optimization of the sigmoidal model if the regularization strength is relatively large. If $\mathcal{L}_G$ is applied less strongly during the earlier iterations of the optimization, then it still is beneficial to perform a warm-up stage, although the performance decrease when not doing so is not as severe, and with more advanced optimization techniques (e.g., strategy 2 or 3) the results become comparable with those for the algorithms with an initial convergence stage. Note that these observations contrast with the conclusion for the sparse baselines, which do not seem to benefit from using a similar unconstrained warm-up stage, as evaluated in Section 3.4.1.

Finally, we perform an ablation study in order to better understand the impact of fine-tuning the learning rate η_G of the logit-activations throughout the optimization procedure. As discussed in Section 3.2.3, η_G is set to 0.5 in the warm-up stage, then decreased to 0.2 for the explore-and-converge stage, and is finally set to 0.1 for the finalization stage.

In this ablation test, η_G is set to 0.2 throughout the entire op-

Table 3.2: Optimization strategy evaluation results: comparison of the metrics evaluated on the outcome of each optimization strategy. For each metric, the mean value over all 135 phrases is shown (standard deviation between parentheses). The results for the peakedness metric for the strong sparsity baseline (*) are computed after discarding any all-zero activation curves in H. For each metric, the most optimal value is shown in bold in each column.

Optimization strategy	**Loss per timestep**			**MAE**	**Overall onset coverage**				**Activations similarity**			**Peakedness**
	$\mathcal{L}_{\text{tot}}/T$	$\mathcal{L}_{\text{KL}}/T$	$\mathcal{L}_G/T$		Pr.	Rec.	F-score	F-score ($\theta_{\text{thr}} = 0.5$)	Min	Mean	Max	Mean
Strategy 0, $\gamma = 1.0$	**0.25**	0.18	**0.07**	0.041	0.82	0.81	0.80	0.71	0.07	0.23	0.56	**0.74**
	(0.07)	(0.05)	**(0.05)**	(0.006)	(0.18)	(0.13)	(0.14)	(0.16)	(0.10)	(0.10)	(0.17)	**(0.05)**
Strategy 0, $\gamma = 0.1$	0.35	**0.12**	0.23	0.035	0.78	0.89	0.82	0.72	0.12	0.31	0.60	0.69
	(0.09)	**(0.02)**	(0.10)	(0.004)	(0.19)	(0.10)	(0.14)	(0.14)	(0.12)	(0.11)	(0.14)	(0.05)
Strategy 1, $\gamma = 1.0$	0.27	0.15	0.12	0.039	0.83	0.82	0.81	0.68	0.08	0.24	0.55	0.70
	(0.07)	(0.04)	(0.06)	(0.005)	(0.18)	(0.13)	(0.13)	(0.16)	(0.09)	(0.09)	(0.17)	(0.04)
Strategy 1, $\gamma = 0.1$	0.41	**0.12**	0.29	**0.034**	0.79	0.90	**0.83**	0.73	0.14	0.32	0.61	0.67
	(0.11)	**(0.02)**	(0.12)	**(0.004)**	(0.18)	(0.09)	**(0.13)**	(0.13)	(0.13)	(0.11)	(0.14)	(0.05)
Strategy 2, $\gamma = 1.0$	**0.25**	0.18	**0.07**	0.042	**0.85**	0.79	0.80	**0.74**	**0.06**	**0.18**	**0.45**	0.71
	(0.06)	(0.04)	**(0.03)**	(0.006)	**(0.17)**	(0.15)	(0.14)	**(0.16)**	**(0.08)**	**(0.09)**	**(0.16)**	(0.07)
Strategy 2, $\gamma = 0.1$	0.34	**0.12**	0.22	0.035	0.79	0.88	0.82	0.73	0.12	0.28	0.56	0.67
	(0.09)	**(0.02)**	(0.09)	(0.004)	(0.19)	(0.11)	(0.14)	(0.13)	(0.11)	(0.10)	(0.15)	(0.06)
Strategy 3, $\gamma = 1.0$	0.27	0.16	0.12	0.039	0.84	0.82	0.82	0.73	0.08	0.20	0.48	0.68
	(0.06)	(0.03)	(0.05)	(0.005)	(0.18)	(0.13)	(0.13)	(0.13)	(0.08)	(0.09)	(0.17)	(0.07)
Strategy 3, $\gamma = 0.1$	0.41	**0.12**	0.30	**0.034**	0.80	0.90	**0.83**	**0.74**	0.14	0.31	0.59	0.66
	(0.12)	**(0.02)**	(0.12)	**(0.004)**	(0.19)	(0.09)	**(0.13)**	**(0.12)**	(0.12)	(0.11)	(0.15)	(0.05)
No normalization of the gradients of G (strategy 2, $\gamma = 0.1$)	0.45	0.17	0.29	0.041	0.64	**0.92**	0.74	0.60	0.11	0.27	0.54	0.71
	(0.19)	(0.20)	(0.14)	(0.02)	(0.17)	**(0.13)**	(0.15)	(0.20)	(0.10)	(0.10)	(0.17)	(0.06)
No warm-up (strategy 0, $\gamma = 1.0$)	2.89	1.29	1.59	0.170	0.66	0.73	0.65	0.23	0.71	0.81	0.90	0.23
	(2.06)	(1.00)	(1.11)	(0.10)	(0.26)	(0.28)	(0.23)	(0.27)	(0.29)	(0.23)	(0.19)	(0.24)
No warm-up (strategy 2, $\gamma = 0.1$)	0.35	**0.12**	0.22	0.035	0.76	0.89	0.81	0.73	0.12	0.29	0.57	0.69
	(0.09)	**(0.03)**	(0.10)	(0.004)	(0.18)	(0.09)	(0.13)	(0.13)	(0.12)	(0.10)	(0.13)	(0.05)
Constant η_G (Strategy 0, $\gamma = 1.0$)	0.28	0.21	0.08	0.044	0.83	0.78	0.79	0.70	0.07	0.23	0.51	0.72
	(0.07)	(0.06)	(0.04)	(0.006)	(0.17)	(0.13)	(0.13)	(0.15)	(0.12)	(0.10)	(0.14)	(0.06)
Constant η_G (Strategy 2, $\gamma = 0.1$)	0.33	0.13	0.20	0.036	0.78	0.87	0.81	0.73	0.11	0.28	0.54	0.69
	(0.09)	(0.02)	(0.09)	(0.004)	(0.18)	(0.10)	(0.12)	(0.12)	(0.12)	(0.11)	(0.13)	(0.07)

timization procedure. This is done for strategy 0 with $\gamma = 1.0$ and for strategy 2 with $\gamma = 0.1$. The former experiment yields an evaluation of the sigmoidal algorithm optimized in a most straightforward way, i.e., without varying learning rates and with the most simple optimization strategy, i.e., strategy 0. Note that this is the simplified algorithm with which the baselines are compared in Section 3.4.1. The results are reported in Table 3.2.

In short, we find that using a more fine-tuned optimization scheme for G is not always effective. For strategy 0 with $\gamma = 1.0$, it leads to slightly lower mean loss per timestep (mean loss per timestep 0.28 without tuning vs. 0.25 with tuning), but to a slightly higher mean loss per timestep for strategy 2 with $\gamma = 0.1$ (mean loss per timestep 0.33 without tuning vs. 0.34 with tuning). We repeated this experiment with the learning rate η_G set to the smaller constant value of 0.1, which performed consistently worse than setting η_G to a constant value of 0.2 (average loss per timestep 0.404 for $\eta_G = 0.1$ vs. 0.33 for $\eta_G = 0.2$ for strategy 2 with $\gamma = 0.1$; average loss per timestep 1.83 for $\eta_G = 0.1$ vs. 0.28 for $\eta_G = 0.2$ for strategy 0 with $\gamma = 1.0$). This illustrates the importance of choosing an appropriate value of η_G in order to ensure a proper convergence of the optimization process.

3.4.3 Example Decomposition

Figures 3.1 and 3.3–3.5 show the decomposition of an example drum loop using, respectively, unconstrained NMFD, sparse NMFD with $\lambda = 0.1$, sparse NMFD with $\lambda = 1.0$, and sigmoidal NMFD (more examples of decompositions are provided as supplementary material to the original publication [1]). We do not show the decomposition using the weak sparsity baseline, $\lambda = 0.01$, as the results are almost identical to those by the unconstrained model. Note that all decompositions reconstruct the spectrogram approximately equally well, except the reconstruction with high sparsity (Figure 3.4).

As mentioned in the introduction (Section 3.1), the activations discovered by unregularized NMFD (Figure 3.1) have two undesir-

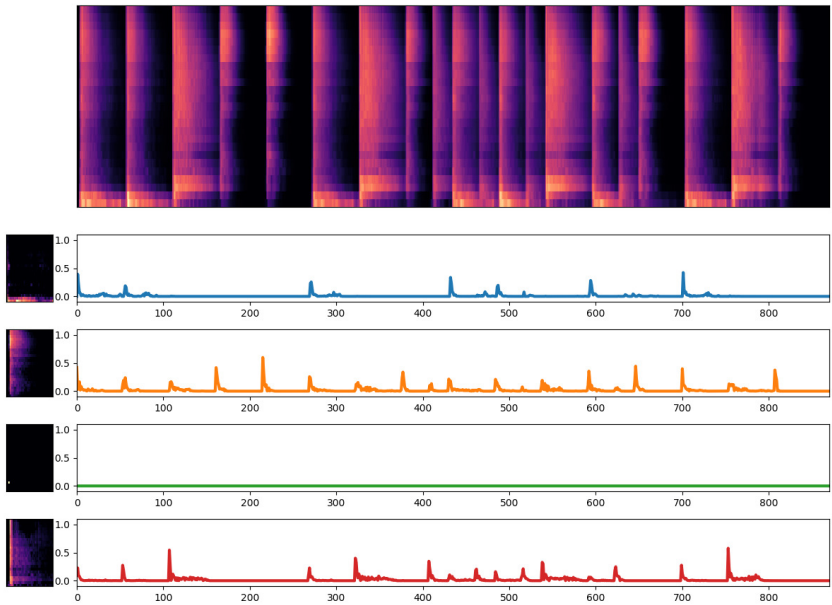


Figure 3.3: Decomposition of a drum loop using sparse NMFD, $\lambda = 0.1$. Although slightly more peaked than the activations for unconstrained NMFD (Figure 3.1), the activations do not show impulse-like behavior, and still contain noisy regions where it is difficult to detect individual drum hit onsets. The third component has become "inactive" in order to minimize the sparsity constraint.

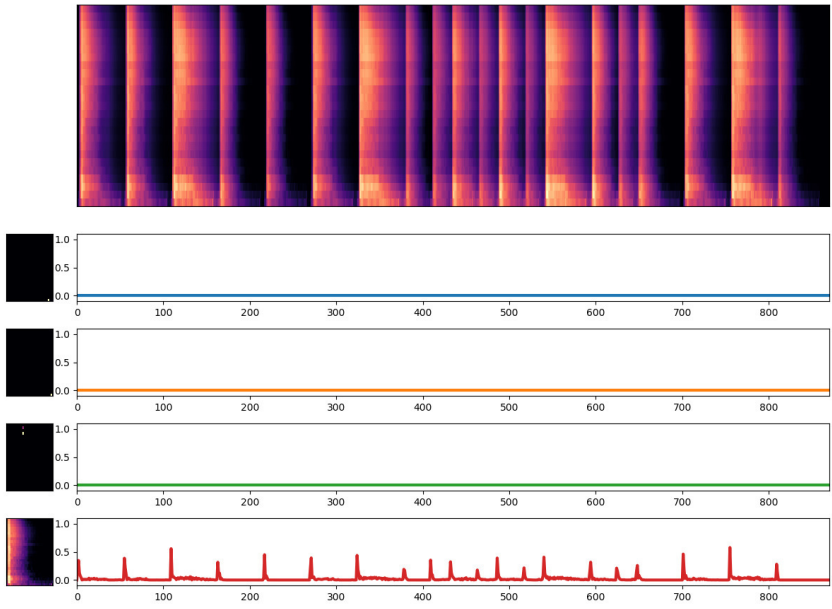


Figure 3.4: Decomposition of a drum loop using sparse NMFD, $\lambda = 1.0$. The decomposition fails because the sparsity constraint is too strong, so that only one component remains active.

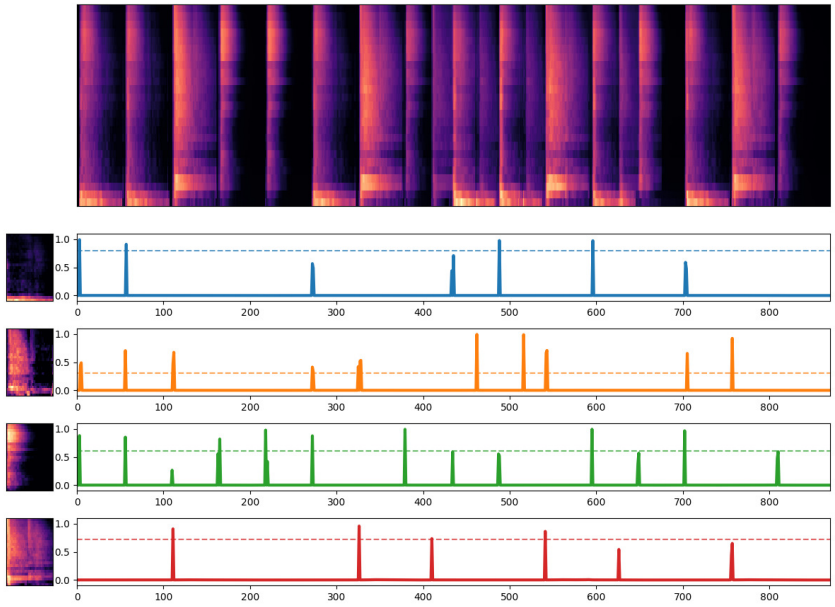


Figure 3.5: Decomposition of a drum loop using sigmoidal NMFD. The activations show impulse-like behavior, and each component captures different parts of the spectrogram, leading to an interpretable decomposition. The dashed line indicates the amplitude a_k for each component.

able properties. The first is that the activations are rather "smeared out", with a sharp initial onset followed by a slowly decaying amplitude. Some small activations are even not preceded by a sharp initial onset, making it hard to determine whether they correspond to a detected drum hit or not. This does not correspond with the expected impulse-like nature of activations of percussive instruments. The second problem is that the activation curves are highly correlated, so that many drum hits are modeled by a mixture of all of the components. This makes it difficult to interpret the resulting decomposition.

When considering the decompositions by sparse NMFD (Figures 3.3 and 3.4), it becomes clear that imposing L1 sparsity does not lead to significantly more impulse-like activations. Even more troublesome is that imposing more sparsity by increasing λ actually hinders a good decomposition: Figure 3.4 illustrates that a high λ causes all but one of the components to become inactive, i.e., their activations are monotonically zero, in order to minimize the (extreme) sparsity constraint as much as possible. This effect is unfortunately also sometimes observed even for reasonable values of λ (Figure 3.3), so that the effective capacity of the NMFD model is reduced in order to minimize the sparsity constraint.

The aforementioned problems are solved by using the sigmoidal NMFD model (Figure 3.5). The activations are highly peaked, and each component models distinct parts of the spectrogram. The allocated capacity, i.e., the number of components K, is used effectively, and it is now very clear where specific sounds are repeated in the mixture. It is worth noting that the proposed regularization term $\mathcal{L}_G$ only provides a direct bias towards binary "on–off" behavior, and that the impulse-like behavior emerges spontaneously when this bias is applied to the decomposition of a percussive mixture. In this example, the components even lend themselves to a musically meaningful interpretation: the first component captures the low end of the kick drum, the second captures the mid- and high-end of the kick drum and snare drum, the third component captures hi-hat hits, and the fourth component models the snare drum

hits. Note that the second component thus contributes to both the kick drum and the snare drum; unfortunately, some entanglement between components is always possible in an unsupervised decomposition. Nevertheless, the decomposition by the sigmoidal model yields much more interpretable results, with activation curves that show to the expected impulse-like behavior.

3.5 Conclusions

In this chapter, we have approached NMFD as an unsupervised decomposition algorithm for percussive music mixtures. Such an unsupervised decomposition is valuable in application scenarios where the exact instruments in the mixture are unknown, or to bootstrap semi-supervised learning approaches such as the one in Wu and Lerch [12]. We investigated an adapted NMFD model where the activations are biased to be binary in nature, by defining them as the output of a sigmoidal function and by applying a regularization term to push their values to saturation. We observe that this results in activations that are highly impulse-like, which correspond to the expected properties of percussive activations, and we have shown that the proposed approach is more effective at obtaining such impulse-like behavior as compared to a sparse NMFD baseline using an L1 sparsity constraint. By means of a case study, we illustrated the potential of our approach to yield more interpretable decompositions.

Regarding future work, we remark that our method, like the original NMFD algorithm, is unsupervised, so that the optimization procedure is free to adapt the templates $W^{(k)}$ without considering their musical validity. Even in an informed setting, where each $W^{(k)}$ is initialized with a template of the desired instrument, there is no guarantee that it will converge to a solution where the components map to individual instruments. This issue could be addressed by adding some kind of supervision to the NMFD framework; this could be a supervised learning algorithm that imposes certain musical constraints learned from data, or an interface where a user

can guide the decomposition interactively. A related direction for further research would be to investigate other and more informed initialization strategies for the templates $W^{(k)}$, and to research how the initialization of the templates impacts the outcome of the optimization process. A second limitation is that this work assumes that the number of components K is known in advance. A next step could therefore be to reliably estimate this number of components prior to decomposition, or to use an iterative decomposition strategy, where K is increased progressively until the full mixture has been decomposed. Finally, we propose that the idea of combining a regularization term that encourages diverging activation values with saturating activations could be incorporated in other models and use cases where binary activations are desired, for example, in the context of music transcription beyond percussive recordings, or even for sound event detection in general.

References

[1] Len Vande Veire, Cedric De Boom, and Tijl De Bie. "Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition". In: *Electronics* 10.3 (2021). ISSN: 2079-9292. DOI: 10 . 3390 / electronics10030284. URL: https://www.mdpi.com/2079-9292/10/3/284.

[2] Tuomas Virtanen. "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria". In: *IEEE Transactions on Audio, Speech and Language Processing* 15.3 (2007), pp. 1066–1074. ISSN: 15587916. DOI: 10.1109/TASL.2006.885253.

[3] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. "Drumkit Transcription via Convolutive NMF". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx 2012), York, UK* 1.3 (2012), pp. 15–18.

[4] Len Vande Veire, Cedric De Boom, and Tijl De Bie. "Adapted NMFD update procedure for removing double hits in drum mixture decompositions". In: *13th International Workshop on Machine Learning and Music at ECML PKDD 2020 (MML2020), Ghent, Belgium* (2020), pp. 10–14.

[5] Paris Smaragdis. "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs". In: *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)* (2004), pp. 494–499. ISSN: 03029743.

[6] Mikkel N. Schmidt and Morten Mørup. "Nonnegative matrix factor 2-D deconvolution for blind single channel source separation". In: *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2006), Charleston, SC, USA* (2006), pp. 700–707. ISSN: 03029743. DOI: 10.1007/11679363_87.

[7] *EDM Drums – Drum Samples Kit by ProducerSpot*. Available online: https://www.producerspot.com/download-free-edm-drums-drum-samples-kit-by-producerspot. (Accessed 02-December-2020).

[8] Olivier Gillet and Gaël Richard. "ENST-Drums: an extensive audio-visual database for drum signals processing". In: *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006), Victoria, Canada* (2006).

[9] Brian McFee et al. "librosa/librosa: 0.8.0 (Version 0.8.0)". In: *Zenodo*. 2020. DOI: http://doi.org/10.5281/zenodo.3955228.

[10] Patricio López-Serrano, Christian Dittmar, Yigitcan Özer, and Meinard Müller. "NMF Toolbox: Music Processing Applications of Nonnegative Matrix Factorization". In: *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), Birmingham, UK* (2019).

[11] Len Vande Veire. *Code for Sigmoidal NMFD: Convolutional NMF with Saturating Activations for Drum Mixture Decomposition*. Available online: https://github.com/aida-ugent/sigmoidal-nmfd (accessed on 12 Jan 2021).

[12] Chih-wei Wu and Alexander Lerch. "From Labeled To Unlabeled Data – on the Data Challenge in Automatic Drum Transcription". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris, France* (2018).

4

Adapted NMFD Update Procedure for Removing Double Hits in Drum Mixture Decompositions

The template length L is an important hyper-parameter in NMFD. It determines the maximum length of the sounds that can be modeled in the framework by one template. Percussive mixtures often contain some instrument(s) with a long decay, such as the kick drum. Therefore, L needs to be large enough to adequately model a single drum hit on these instruments.

Unfortunately, when the template length is relatively long, then NMFD has a tendency to capture multiple drum hits within one template. This issue has been noted before in the context of drum mixture decomposition using NMFD [1, 2]. It usually occurs when the percussive mixture contains both drum hits with a long decay, e.g. a kick drum, as well as hits that follow each

other in rapid succession, e.g. hi-hats. Note that the excess hits in a template can be either of the same instrument as the first hit, e.g. two hi-hat hits captured within one template, but that they can also be arbitrary drum sequences of different instruments. In both cases, the occurrence of multiple hits in a template is problematic: the resulting activations then no longer reflect the onsets of individual instruments, making the decomposition less interpretable and useful. Figure 4.1(b) illustrates this problem.

One solution to this problem, as proposed by Lindsay-Smith et al. [1], is to initialize $W^{(k)}$ with a prototypical sample of the drum sounds that one expects to find within the mixture, and then keep the templates fixed throughout optimization. However, this limits the application of NMFD to the transcription of mixtures where the source sounds are known or can be estimated sufficiently well before decomposition. Becker et al. [2] propose to give each template $W^{(k)}$ a different length that is as large as necessary, but as small as possible for that particular sound source. These templates with an adaptive length are constructed by first performing NMF on the spectrogram, and then iteratively merging components with similar activation patterns.

In this chapter, I propose a more direct approach, where the emergence of "double hits" is detected during optimization. As soon as they start to emerge, they are replaced with an exponentially decaying extrapolation of the preceding template frames. I demonstrate the effectiveness of this approach with an experiment on a large-scale dataset of solo drum recordings.

> *The research presented in this chapter has been presented at the 13th International Workshop on Machine Learning and Music (MML) at the ECML-PKDD 2020 conference. The corresponding publication is:*
>
> *Vande Veire L., De Boom C., De Bie T. Adapted NMFD update procedure for removing double hits in drum mixture decompositions. In 13th International Workshop on Machine Learning and Music at ECML-PKDD 2020 (MML2020), Ghent, Belgium, 2020, pp. 10-14 [3] .*

4.1 Detecting emerging double hits during optimization

I propose to solve the 'double-hit' problem by checking after each update of $W^{(k)}$ whether a second onset can be detected in the template. If this is the case, then $W^{(k)}$ is modified by overwriting this second onset with an exponentially decaying extension of the preceding template frames. This will initially lead to a worse approximation of the spectrogram, as important information for the decomposition was removed. However, the expected effect of this modification is that, in the next update of the activations H, some activation value(s) will increase to compensate for the removal of the secondary onset in the template; eventually, after a few updates, each $W^{(k)}$ will ideally only contain a single drum hit, and all onsets will be captured in H_k.

The adapted update procedure for $W^{(k)}$ is as follows:

1. Calculate the updated version of $W^{(k)}$, as in Eqn. (2.4).

2. Calculate the log-envelope $a^{(k)}[\tau]$ of each updated template $W^{(k)}$:

$$\tilde{a}^{(k)}[\tau] = \sum_n \log\left(W^{(k)}_{n,\tau} + \epsilon\right), \tag{4.1}$$

$$a^{(k)}[\tau] = \tilde{a}^{(k)}[\tau] - \min_\tau\left(\tilde{a}^{(k)}[\tau]\right). \tag{4.2}$$

3. Calculate $\Delta a^{(k)}[\tau] = a^{(k)}[\tau + \tau_u] - a^{(k)}[\tau]$. When $\Delta a^{(k)}[\tau]$ is large for some τ, then there is an onset at time τ in the template.

4. Set $a^{(k)}_{\max} = \max\left(a^{(k)}[\tau]\right)$. Detect onsets in $W^{(k)}$ by determining whether there is an onset larger than some threshold θ_{thr},

$\Delta a^{(k)}[\tau] \geq (\theta_{\text{thr}}\, a^{(k)}_{\text{max}})$, for some $\tau \geq \tau_{\text{thr}}$. Only peaks that lie past the shift threshold τ_{thr} are considered, in order to not erroneously correct the first (and correct) hit in the template.

5. If there is a second onset in the template at $\tau_{\text{err}} \geq \tau_{\text{thr}}$, then all the frames after this onset are replaced by an exponentially decaying extension of the template frames preceding it:

$$W^{(k)}_{n,\tau} \leftarrow W^{(k)}_{n,\tau_{err}-\tau_u} \exp\left(-\gamma(\tau - \tau_{\text{err}})\right), \tau = \tau_{\text{err}} \ldots L_\tau. \quad (4.3)$$

In our experiments, we use the following settings for the hyper-parameters of this procedure: $\tau_u = 3$, $\theta_{\text{thr}} = 0.05$, $\tau_{\text{thr}} = 10$, $\gamma = 1$, $L_\tau = 50$, $\epsilon = 10^{-18}$, which were empirically found to lead to good results. The STFT spectrogram is calculated with a hop size of 512, and the audio sampling rate is 44.1 kHz.

4.2 Case study: decomposing a drum loop

As an example, we consider the drum loop in Figure 4.1(a)[1]. It contains three instruments: a kick drum, a snare drum and a hi-hat. The kick drum decays over approximately 50 frames; hence, we set $L_\tau = 50$. We note, however, that the hi-hats occur in rapid succession, i.e. approximately every 25 frames.

When decomposed with the original NMFD algorithm, shown in Figure 4.1(b), the templates $W^{(k)}$ capture not the individual drum hits, but rather repeating *sub-sequences* of drum hits. The activations consequently are very sparse and are not informative to determine the onset locations of the individual instruments.

When decomposed with NMFD using the proposed modifications, the templates each capture only a single drum hit, as shown in Figure 4.1(c). Note that the extracted templates very much resemble their ground-truth counterpart, see Figure 4.1(a). The activations also match the ground-truth onsets quite well; for the

[1]This drum loop is a 4 second extract of a solo drum recording from the ENST dataset [4], *"062_phrase_rock_simple_medium_sticks.wav"*.

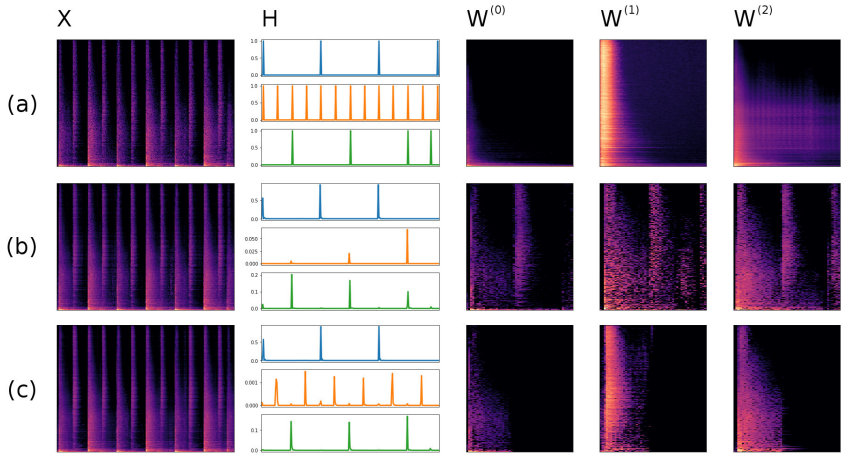


Figure 4.1: Illlustration of the decomposition of a short drum loop: (a) ground-truth decomposition; (b) decomposition with NMFD; (c) decomposition with the modified NMFD algorithm. Columns: X, the spectrogram; H, the activations; $W^{(0)}$, the first template, capturing the kick drum; $W^{(1)}$, capturing the hi-hats; $W^{(2)}$, capturing the snare drum.

hi-hat, i.e. the second component, there is some discrepancy, as only every other onset is clearly captured. The other activations are 'absorbed' into the kick drum and snare drum components. This is a consequence of the fact that NMFD cannot distinguish a single-instrument hit from such a consistent layering of multiple instantaneous drum hits (i.e. in this example, each kick/snare drum hit always coincides with a hi-hat hit); an additional mechanism to disentangle such sounds is beyond the scope of the work presented in this chapter.

4.3 Evaluation on the ENST dataset

We evaluate our approach on all *fast simple* phrases from the ENST dataset [4]. We run the original NMFD algorithm and our adaptation on these extracts, and quantify how many excess drum hits can be detected in each template by counting the number of peaks in $\Delta a^{(k)}[\tau]$, see Section 4.1. We furthermore measure the spectrogram reconstruction quality using the Mean Absolute Error between X and $\hat{X}$: $\text{MAE}(X, \hat{X}) = \frac{1}{NT} \sum_{n,t} |X_{n,t} - \hat{X}_{n,t}|$.

For each decomposed mixture, the MAE for the decomposition with the original algorithm and the MAE for the adapted version are nearly identical; furthermore, all spectrograms are approximated well (mean MAE $5.6 \cdot 10^{-5}$ for both the original and the adapted algorithm, stdev. $3.0 \cdot 10^{-5}$ and $3.1 \cdot 10^{-5}$ resp.). The average number of excess peaks detected in $\Delta a^{(k)}[\tau]$ is 2.2 (stdev. 1.0) for default NMFD, and 0 for the adapted procedure[2]. Visual inspection[3] of the results shows that in the decompositions with unmodified NMFD, double hits are often present, while these are removed with the proposed procedure.

4.4 Conclusions

We conclude that the proposed adaptation maintains the same spectrogram reconstruction quality, with the added advantage that NMFD now captures only one drum hit per template. This allows to choose the template length long enough to fully capture drum hits with a long decay, while maintaining a clear and interpretable decomposition even in the presence of rapid successive drum hits.

[2]Which is an expected result, of course, as we report on the metric that is used in the adapted algorithm to detect double hits in the templates.

[3]See the accompanying website for examples: `https://lenvdv.github.io/papers/2020-nmfd-double-hit-examples/`

References

[1] Henry Lindsay-Smith, Skot McDonald, and Mark Sandler. "Drumkit Transcription via Convolutive NMF". In: *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx 2012), York, UK* 1.3 (2012), pp. 15–18.

[2] Julian M. Becker and Christian Rohlfing. "Custom sized non-negative matrix factor deconvolution for sound source separation". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)* 3 (2014), pp. 2124–2128. ISSN: 15206149. DOI: 10.1109/ICASSP.2014.6853974.

[3] Len Vande Veire, Cedric De Boom, and Tijl De Bie. "Adapted NMFD update procedure for removing double hits in drum mixture decompositions". In: *13th International Workshop on Machine Learning and Music at ECML PKDD 2020 (MML2020), Ghent, Belgium* (2020), pp. 10–14.

[4] Olivier Gillet and Gaël Richard. "ENST-Drums: an extensive audio-visual database for drum signals processing". In: *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006), Victoria, Canada* (2006).

Part II

Analyzing and Manipulating Drum'n'bass using Music Information Retrieval Techniques

5

From Raw Audio to a Seamless Mix: Developing an Automated DJ System for Drum'n'bass Music

This chapter presents the open-source implementation of the first fully automatic and comprehensive DJ system, able to generate seamless music mixes using songs from a given library much like a human DJ does. The proposed system is built on top of several enhanced music information retrieval (MIR) techniques, such as for beat tracking, downbeat tracking and structural segmentation, to obtain an understanding of the musical structure. Leveraging the understanding of the music tracks offered by these state-of-the-art MIR techniques, the proposed system surpasses existing automatic DJ systems both in accuracy and completeness. To the best of our knowledge, it is the first fully integrated solution that takes all basic DJing best practices into account, from beat and downbeat matching to identification of suitable

cue points, determining a suitable cross-fade profile and compiling an interesting playlist that trades off innovation with continuity.

To make this possible, I focus on one specific subgenre of electronic dance music, namely drum'n'bass. This allows to exploit genre-specific properties, resulting in a more robust performance and tailored mixing behavior. Evaluation on a corpus of 160 drum'n'bass songs and an additional hold-out set of 220 songs shows that the used MIR algorithms can annotate 91% of the songs with fully correct annotations (tempo, beats, downbeats, and structure for cue points). On these songs, the proposed song selection process and the implemented DJing techniques enable the system to generate mixes of high quality, as confirmed by a subjective user test in which 18 drum'n'bass fans participated.

> *The research presented in this chapter has been published in the EURASIP Journal on Audio, Speech, and Music Processing. This publication is the result of continued work on my Master's thesis, more specifically in terms of the track selection component of the automated DJ (including the development and incorporation of the musical style descriptor and a vocal clash detection system) and the evaluation of the DJ system in a user study. The corresponding publications are:*
>
> *Vande Veire L., De Bie T. From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass. In Journal on Audio, Speech, and Music Processing. 2018, 13, 2018.* DOI: *10.1186/s13636-018-0134-8.* URL: *https://doi.org/10.1186/s13636-018-0134-8 [1] .*
>
> *Vande Veire L. From raw audio to a seamless mix: an Artificial Intelligence approach to creating an automated DJ system. Master's thesis, Ghent University, 2017.* URL: *https://lib.ugent.be/catalog/rug01:002367502 [2].*

5.1 Introduction

When music tracks are played back to back, i.e. starting one song after the other is finished, the listening experience is interrupted between the end of a song and the beginning of the next. Indeed,

popular music tracks commonly have a long intro and outro, such that the excitement of listening might fade if songs are played in full. Thus, especially for electronic dance music (EDM) in dance clubs, it is common practice for so-called *Disk Jockeys (DJs)* to blend songs together into a continuous seamless mix.

Unfortunately, DJing requires considerable expertise, specialized equipment, and time – unavailable to most music consumers. A computer program that automates the DJing task would thus democratize access to high-quality continuous music mixes outside the dance club setting. Additionally, it would alleviate the need for nightclubs and bars with a limited budget to hire expensive human DJs. Finally, professional DJs could use it as an exploration tool to discover interesting song combinations.

As DJing is a complex analytical as well as creative task, creating an automatic DJ has proved to be highly non-trivial (see Sec. 5.2). To clarify the challenges involved, next we will discuss what a DJ is and what steps are performed to create a music mix.

5.1.1 What is a DJ?

A DJ is a person who mixes existing music records together in a seamless way to create a continuous stream of music [3–5]. With a *seamless* mix, we understand a mix that blends songs together such that the resulting music is uninterrupted (no silences in between songs), and such that the music is structurally coherent on *beat*, *downbeat* and *phrase* level (see Sec. 5.3.1). Additionally, successive songs should be 'compatible' to some extent with respect to their harmonic, rhythmic and/or timbrical properties. In essence, a seamless mix flows from song to song such that the transition between those songs appears to be a part of the music itself, and where it consequently is often hard to tell where one song ends and the other begins. Even though there is no step-by-step "recipe" on how to DJ, there is a general consensus [3, 4] on the basic steps that the DJ executes to create a mix. As a brief introduction to the art of DJing, a simplified DJing workflow is explained below

and illustrated in Figure 5.1.

Creating a mix The DJ first selects songs to play and determines the order to play them in. This is the *track selection* step. The DJ selects songs that fit together thematically, rhythmically or instrumentally, while also considering the audience's engagement, music preference, and other factors [3–5]. There usually is a deliberate progression throughout the mix [5, pp. 311, 328–329]: for example, the DJ may start with a more relaxed song, gradually building up the energy by successively playing increasingly energetic songs. After reaching the climax, the DJ may play some calmer songs to give the crowd a rest, before building up towards a second climax, etc. The DJ also determines at what time instants in the selected songs the transition from one song into the next should start, which is called *cue point selection*. These starting points, or *cue points*, are typically aligned with structurally important boundaries in the music [3–5]: this ensures that the mix forms a contiguous piece of music that naturally "flows" from one song into another [5, pp. 316–318].

With the songs and cue points in mind, the DJ performs the actual mixing. He or she plays the first song and waits until it reaches the cue point to start the second song. For some time both songs will be heard simultaneously, gradually fading out the first song and fading in the next. When simultaneously playing two songs, it is imperative that their beats align in time: even a very small misalignment is easily noticed even by the amateur. To make this possible, one or both songs may have to be slowed down or sped up by *time-stretching* them such that their tempi are equal. The beats are then aligned in a process called *beatmatching*.

A smooth transition between the songs is established by performing a *crossfade*. This is the process of gradually increasing the volume of the new song, i.e. the fade-in, while decreasing the volume of the other song, i.e. the fade-out. The DJ also adjusts the *equalization* settings of the songs by adjusting the relative gain of

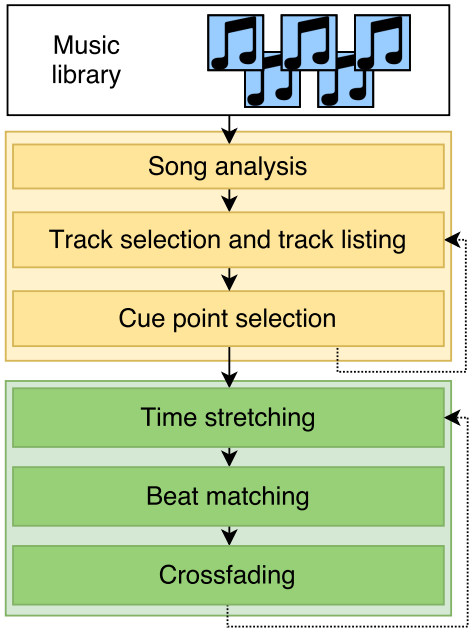


Figure 5.1: Illustration of the simplified DJing workflow.

the bass (low frequencies), mid and treble (high frequencies): this ensures a clean sound of the mix without saturation in any of the frequency bands. Effectively, this means that the speed and profile of the crossfade may be different for different frequency bands.

The process of cueing, time-stretching, beatmatching and crossfading is applied for each song transition, effectively creating a seamless music stream.

Remarks on the DJing process It should be pointed out that the workflow described above is a simplification and not an exact step-by-step recipe of the DJing workflow. For conciseness, some aspects of the DJing process have not been elaborated on in detail. For example, depending on e.g. the audience or type of event, DJs might employ a different mixing style and consider some aspects of

the process (e.g. correct beatmatching) to be less or more important than other DJs in other scenarios [4]. It should also be clear that DJing is an iterative process, i.e. the steps shown in Figure 5.1 are often repeated and interwoven as the mix progresses. For example, improvisation plays an important role in DJing [5, p. 312] and DJs don't always know beforehand which songs they will play: the track selection and cue point selection process are hence repeated "on the fly" while the DJ is mixing. For a more thorough introduction into the art of DJing, we refer to relevant works in non-academic [3, 4] and academic literature [5].

Understanding musical structure It will be clear that, in order to perform the above tasks, the DJ first and foremost needs to know the tempo, beat positions, structure and other musical properties as discussed below in Sec. 5.3.1.

Algorithms have been developed for each of the tasks of interest to us in this chapter. However, the specific application in this chapter poses specific demands on accuracy and robustness, while also offering opportunities for improvement. For example, DJing is almost invariably done with songs with a steady tempo [5, pp. 9, 33–34] (expressed in *beats per minute*), making the task of tempo estimation easier than for other types of music. At the same time, the required tempo estimation accuracy exceeds the accuracy of state-of-the-art tempo estimation algorithms (see Sec. 5.4.2).

5.1.2 Contributions and overview

In this chapter, we present a computer system that automates the tasks of a DJ. More specifically, we make the following contributions:

- The system is, to the best of the author's knowledge, the first complete and fully integrated automatic DJ system that considers all basic DJing best practices, i.e. beatmatching, cue

point selection, music style-based track selection, equalization and different transition types. It is released as open-source[1] and could hence serve as a robust basis for further research to build upon.

- The system is designed for a specific genre of electronic dance music, namely drum'n'bass. Apart from simplifying system design and dataset collection, this allows it to obtain a high structural annotation accuracy and an excellent subjective mix quality, because prior knowledge of the genre allows to exploit certain assumptions about the musical structure (see Sec. 5.3.1). For this system, drum'n'bass was chosen given the author's knowledge of the genre.

- To achieve this performance, dedicated algorithms for tempo estimation, beat detection, downbeat detection, and cue point selection were developed.

- A unique feature of our proposed system is a song selection method that imitates the behavior of a professional DJ. It uses a custom *style descriptor* feature, that projects all songs into a continuous "style space" where similar songs lie close together. This approach greatly improves the mix quality.

The remainder of this chapter is structured as follows. Section 5.2 explores related work in scientific literature and in commercial applications. Section 5.3.1 elaborates on how the automatic DJ discovers the musical structure in a hierarchical manner. Section 5.3.2 discusses the system architecture, the song selection process and how the song transitions are created. The system's

[1] The code for the DJ system is made available as open-source and can be downloaded from `https://bitbucket.org/ghentdatascience/dj` under an AGPLv3 license. An updated version, featuring restructured and more modular code in Python 3, a more consistent storage of song annotations, and stereo audio support, can be found at `https://www.github.com/lenvdv/dnb-autodj-3`.

performance is thoroughly evaluated on different aspects in Section 5.4. Finally, Section 5.5 concludes this chapter and gives some pointers for further improvements.

5.2 Related work

Existing research on automatic DJ systems is scarce. Two types of systems reoccur in the scientific literature: automatic DJ systems and mash-up systems. The former attempt to automate (parts of) the DJing task, i.e. create a continuous mix by smoothly transitioning between songs. Mash-up systems on the other hand create a new song by combining short fragments of existing songs. A mash-up is typically much shorter than a DJ mix, and the input songs are more heavily modified by cutting and pasting fragments from them. Nevertheless, similar techniques, such as time-stretching, beat tracking and crossfading, are used in both applications.

Jehan [6] proposes a simple automatic DJ system that automatically matches downbeats and crossfades songs. Cue points are determined by finding rhythmically similar sections in the mixed songs, but it does not consider the high-level structure of the music. Bouckenhove [7] describes a system that uses vocal activity detection to avoid overlapping vocal sections of two songs. With their Music Paste system, Lin et al. [8] automate the track and cue point selection process by maximizing a measure for musical similarity. The length of the transition is optimized such that the rate of tempo change remains under an acceptable threshold, which is determined in a subjective experiment. Ishizaki et al. [9] also focus on the optimization of a crossfade with a changing tempo. They propose to use an intermediary tempo in between the tempi of the original songs, such that the discomfort caused by the tempo change is spread evenly between the two songs. Finally, the MusicMixer project by Hirai et al. [10] improves the track and cue point selection process by means of two similarity

Table 5.1: Overview of existing (automatic) DJ software and related work.

	Type (Mashup, DJ, Other)	Autonomous	Beat tracking	Downbeat tracking	Structural segmentation	Time-stretching/beatmatching	Key detection/Harmonic mixing	DJ-inspired track selection	Different transition styles	Audio equalization	Vocal activity detection	Rhythmic compatibility	Open-source
Jehan [6]	DJ	✓	✓	✓	±	✓						✓	
Bouckenhove [7]	DJ	✓	✓			✓	✓		✓		✓		
Music Paste [8]	DJ	✓	✓			✓	✓					✓	
Ishizaki et al. [9]	DJ	✓	✓			✓							
MusicMixer [10]	DJ	✓										✓	
AutoMashUpper [11]	M	✓	✓	✓	✓	✓	✓					✓	
Lee et al. [12]	M	✓	✓			✓	✓						
Mixed In Key [13]	O		✓	✓	✓	*n/a*	✓	*n/a*	*n/a*	*n/a*		*n/a*	
Mashup2 [14]	M		✓	✓		✓	✓			✓			
Serato DJ [15]	DJ		✓	✓		✓	✓	*n/a*	*n/a*	✓			
Traktor Pro 2 [16]	DJ	±[a]	✓	✓		✓	✓			✓			
Virtual DJ [17]	DJ	±[a]	✓	✓		✓	✓			✓			
Mixxx [18]	DJ	±[a]	✓	✓		✓	✓			✓			✓
Serato Pyro [19]	DJ	✓	✓	?[b]	?	✓	?	?	?	?	?	?	
Pacemaker [20]	DJ	✓	✓	?	?	✓	?	?	?	?	?	?	
Our DJ system	**DJ**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

[a] Even though this software has many features that can be used by a human DJ (e.g. beat and downbeat tracking, key detection, equalization), most of these features appear to be unused in the software's automatic DJ mode, which remains rather simple.

[b] Implementation details of this software are unknown, and the product specifications do not explicitly mention these capabilities. Hence, it is unclear whether the software implements any of the properties marked with "?".

measures related to the beat structure and a high-level abstraction of the chromatic content of the audio, inspired by natural language processing techniques.

Research on mash-up systems typically focuses on devising a measure of musical compatibility of song extracts. An example is the AutoMashUpper system by Davies et al. [11, 21], which features a beat tracking, downbeat tracking and a structural segmentation step to align the music. Music fragments are extracted based on their harmonic and rhythmic similarity. Lee et al. [12] focus on extending mash-up systems to multiple overlapping songs and also consider the compatibility of consecutive music segments instead of only the compatibility of the overlapped segments.

Most existing work on automatic DJ systems focuses on optimizing individual crossfades by minimizing the amount of discomfort. However, there are still some important limitations to these systems. First of all, the crossfading process often remains very simple, e.g. without performing any equalization. Secondly, very few systems consider the high-level structural properties of the mixed audio. Thirdly, the focus is usually on optimizing individual crossfades, but the global song progression throughout the mix is not considered. In general, there appears to be no complete integrated DJing system in scientific literature that elegantly combines all necessary components and considers DJing best practices to create enjoyable music mixes.

There also exist many commercial DJing applications. Examples of professional DJing software include Serato DJ [15] and Traktor Pro 2 [16] or free alternatives such as VirtualDJ [17] and Mixxx [18, 22]. These aid the DJ in analyzing the music by annotating e.g. the tempo, beat and downbeat positions and the musical key. Most of this software is tailored to DJs who perform the mixing themselves using advanced DJing equipment such as turntables and mixers, but some feature automatic DJing

functionality as well. Another application designed for DJs is Mixed In Key [13], which aids DJs in performing harmonic mixing. It is not DJing software itself, but rather an annotation tool that extracts a song's key, tempo, relevant cue points and a custom "energy level" annotation. Mixed In Key can be integrated with DJing software such as Serato DJ or Traktor Pro 2. The company behind Mixed In Key also released Mashup2, a mashup creation program that features automatic beat matching and key compatibility detection [14]. There also exist apps whose main purpose is to automate the task of a DJ. Examples include the mobile apps Serato Pyro [19] and Pacemaker [20]. However, the automatic DJ capabilities in commercial applications remain quite simple. Typically, the transitions follow a basic "intro-outro" paradigm, where the next song is played only when the current song ends [23, 24]. Informal experimentation furthermore indicates an inferior performance in terms of beat detection accuracy and no or only very basic cue point selection capabilities. Finally, none of the aforementioned commercial applications are open-source or explain the inner workings of their algorithms. Hence, to the best of our knowledge, no well-documented, open-source automatic DJ solution exists that combines existing MIR knowledge and DJing best practices as in the proposed project.

A common trend in existing work is to deal with a broad variety of genres. However, the presented system is designed to explicitly deal with only one genre, namely drum'n'bass. In their *One In The Jungle* paper, Hockman, Davies and Fujinaga [25] already point out the need for genre-specific approaches within MIR research, more specifically for genres like drum'n'bass that have e.g. very distinct drum patterns. The proposed system gives further proof that a genre-specific approach might indeed be beneficial for certain applications.

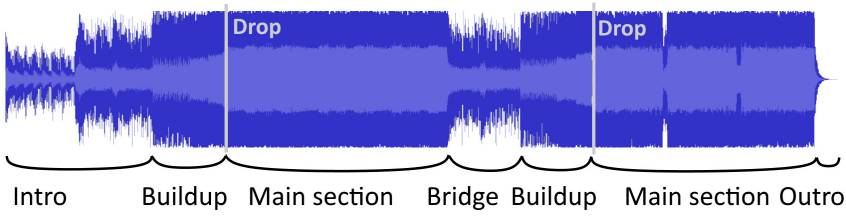


Figure 5.2: Example of the compositional layout of a drum'n'bass song [5, pp. 287–292]. A song typically starts with an *intro*, followed by a *build-up* that gradually increases the musical tension. This tension is released in a moment called the *drop*, which is the beginning of the *main section* or *core* [5, p. 289] of the music, comparable to the chorus in pop music. After the main section, there is a musical *bridge*, also called a *breakdown*, that connects the first main section and the second build-up, drop and main section. An *outro* gradually brings the music to a close. Note that this structure is not fixed and that many variations are of course possible.

5.3 Methods

5.3.1 Discovering the musical structure

One characteristic of music as an audio signal is that it exhibits a hierarchical structure [5, 26]. At the lowest level, music consists of individual musical events or notes, which repeat periodically to define the tempo of the music. Certain events (typically percussive in nature) are more prominent than others, creating the *beats*. In drum'n'bass, like in most types of EDM, beats can be grouped into groups of four [5, pp. 246–248], which are called *measures* or *bars*. The first beat of a measure is called the *downbeat* of that measure. Measures are the basic building blocks of longer musical structures such as *phrases*, which then make up the larger *sections* that determine the compositional layout of the song. The typical composition of a drum'n'bass song is illustrated in Figure 5.2.

Knowing the rhythmical and structural properties of the music is extremely important for a DJ. The tempo and beat positions are used to *beatmatch* the music. The high-level musical structure is important when selecting cue points: if the DJ mixes songs where the downbeats or segment boundaries are not appropriately aligned, the mix will most likely sound structurally incoherent[5, pp. 317–318].

The automatic DJ system is designed for a specific music genre, and it therefore makes several assumptions about the music's structural properties:

- the music's tempo is between 160 and 190 beats per minute and is assumed to be constant throughout the song,
- the music has a strict 4/4 *time signature*, i.e. there are four beats to a bar,
- phrases are a multiple of 8 measures long, and musical segments are an integer number of phrases long. Boundaries between segments are accentuated by large changes in musical texture.

Even though these assumptions might seem restrictive, they are based on extensive experience with the target music genre and should hold for a vast majority of drum'n'bass songs. Some of these properties have also been noted in other works for EDM in general [5, pp. 246–248]. Furthermore, we believe that it is feasible to adapt the mentioned techniques to different EDM genres (see Sec. 5.5).

In what follows is described how the proposed DJ system extracts the beat, downbeat and segment boundary locations from the audio in a hierarchical way given the assumptions listed above.

Beat tracking

To discover the beat positions, an algorithm inspired by the work of Davies and Plumbley [27] is used. It assumes a constant tempo, which is the case for the vast majority of drum'n'bass music. With this assumption, only two parameters need to be determined to define the positions of all the beats: the beat period τ (expressed in seconds) or equivalently the tempo $\nu = \frac{60}{\tau}$ (expressed as 'beats per minute'), and the beat phase ϕ (expressed in seconds), i.e. the time difference between the first beat and the start of the audio signal. Two observations are at the core of the beat tracking algorithm. Firstly, most repetitions of musical *onsets*, e.g. played notes or percussion events, happen after an integer number of beats. Secondly, the loudest or most prominent onsets typically occur on beat locations. To exploit these observations, the positions of musical onsets are estimated by means of an *onset detection function* (ODF), which has a high value for time instants in the music where an onset is detected, and a low value elsewhere. An excellent review on the different types of onset detection functions is given by Bello et al. [28].

Figure 5.3 illustrates the beat tracking algorithm. The first step is to calculate the ODF $\Gamma(m)$ of the audio. The *melflux* ODF is used [29] with a frame size N_F of 1024 samples and hop size N_H of 512 samples. The audio sample rate f_s is 44100 Hz. With the notations explained in Table 5.2, $X_{\text{mel40}}(m, k)$ being the energy of frame m in the k^{th} frequency bin, logarithmically spaced according to the Mel scale [30], and HWR the half-wave rectify operation $\text{HWR}(x) = \max(x, 0)$, the ODF is calculated as follows:

$$\Gamma(m) = \sum_{k=1}^{40} \text{HWR}(X_{\text{mel40}}(m, k) - X_{\text{mel40}}(m-1, k)). \qquad (5.1)$$

This curve is post-processed by subtracting a running mean with

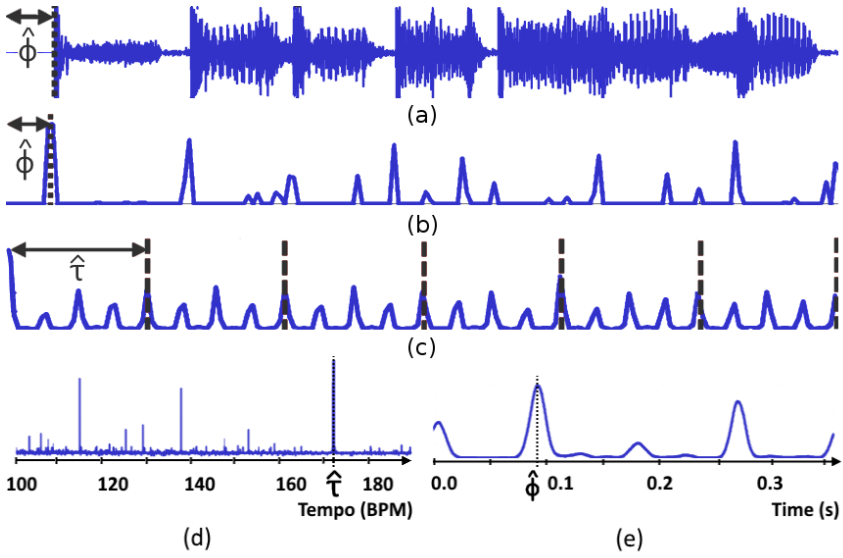


Figure 5.3: Illustration of the beat tracking algorithm. (a) Audio waveform extract. (b) Half-wave rectified onset detection function $\Gamma_{\mathrm{HWR}}(m)$. (c) Autocorrelation function $A_{\Gamma}(l)$ of the ODF. (d) Tempo detection curve $\mathcal{B}(\tau)$. (e) Phase detection curve $\Phi(\phi)$. Our algorithm's estimate of the beat period $\hat{\tau}$ and the beat phase $\hat{\phi}$ are annotated. This figure was created by applying the algorithm to a piece of audio of 4m30s long. For illustrative purposes, only the first few seconds of the audio waveform, ODF and autocorrelation function are shown. Note that the described algorithm detects the beats between 160 and 190 BPM, and a larger tempo range is shown in (d) for illustrative purposes.

Table 5.2: Overview of mathematical notations used in this Chapter.

Notation	**Explanation**
$x(n)$	Audio signal, sample n.
$x_{\text{frame}}^{[N_F;N_H]}(m)$	Audio frame m using frame size N_F samples and hop size N_H samples.
$X^{[N_F;N_H]}(m,k)$	Spectrum or spectrum-like features of frame m, frequency bin k, calculated on audio frames using frame size N_F samples and hop size N_H samples.
$\mathcal{X}^{[N_F;N_H]}(m[,k,l,...])$	Feature vector for frame m, calculated on audio frames using frame size N_F samples and hop size N_H samples. Additional indices k or l are used to index subcomponents of the feature vector.
$\Gamma(\cdot)$	Onset detection function.
$N_t = f_s t$	The number of samples within a time difference t.
$L_t = \frac{N_t}{N_H}$	The number of frames within a time difference t.
$\lfloor \cdot \rceil$	Rounding operator (round to the nearest integer)

window size Q frames from it and half-wave rectifying the result:

$$\bar{\Gamma}(m) = \underset{m-\frac{Q}{2} \leq q \leq m+\frac{Q}{2}}{\text{mean}} \{\Gamma(q)\}, \tag{5.2}$$

$$\Gamma_{\text{HWR}}(m) = \text{HWR}\left(\Gamma(m) - \bar{\Gamma}(m)\right). \tag{5.3}$$

Q is set to 16 ODF frames, as in [27]. Then the beat period is extracted by calculating the autocorrelation function $A_\Gamma(l)$ of the ODF:

$$A_\Gamma(l) = \frac{1}{M} \sum_m \Gamma_{\text{HWR}}(m)\Gamma_{\text{HWR}}(l+m). \tag{5.4}$$

The autocorrelation $A_\Gamma(l)$ is large at lag values l that are a multiple of the beat period τ. Thus, we propose to estimate the beat period τ as the one for which the sum of autocorrelation values at corresponding lags (integer multiples of τ) is maximal. Concretely, we define the *tempo detection curve* $\mathcal{B}(\tau)$ as:

$$\mathcal{B}(\tau) = \frac{1}{N} \sum_n A(n\, L_\tau), \tag{5.5}$$

and obtain an estimate $\hat{\tau}$ for the beat period as:

$$\hat{\tau} = \underset{\tau}{\text{argmax}}(\mathcal{B}(\tau)), \tag{5.6}$$

with a corresponding lag value $L_{\hat{\tau}}$ (in frames). Note that $L_{\hat{\tau}}$ does not need to be an integer. The beat phase is determined by summing the ODF values for every possible time shift ϕ between 0 and $\hat{\tau}$, at fixed intervals of one beat period. The phase is then estimated as the one leading to the highest sum. Formally, defining the *phase detection curve* as:

$$\Phi(\phi) = \frac{1}{N} \sum_n \Gamma_{\text{HWR}}(n\, L_{\hat{\tau}} + L_\phi), \tag{5.7}$$

the phase is estimated as:

$$\hat{\phi} = \underset{\phi}{\text{argmax}}(\Phi(\phi)). \tag{5.8}$$

The estimated position of the m'th beat is (with $\lfloor \cdot \rceil$ rounding to the nearest integer):

$$t_{\text{beat}}^{(m)} = m\hat{\tau} + \hat{\phi} \qquad \text{(in seconds)}, \tag{5.9}$$

$$L_{\text{beat}}^{(m)} = \lfloor mL_{\hat{\tau}} + L_{\hat{\phi}} \rceil \qquad \text{(in frames)}, \tag{5.10}$$

$$N_{\text{beat}}^{(m)} = \lfloor mN_{\hat{\tau}} + N_{\hat{\phi}} \rceil \qquad \text{(in samples)}. \tag{5.11}$$

The tempo range is restricted between 160 BPM and 190 BPM ($160 \leq \nu \leq 190$), because the tempo of drum'n'bass music falls between these values. We consider increments of 0.01 BPM for the tempo and $1ms$ for the phase.

Downbeat tracking

Given the beat positions, the proposed DJ system determines which beats are downbeats. Since measures of 4 beats long are assumed, there are only four options: the first downbeat of the song is either the first, the second, the third or the fourth beat, and every fourth beat after that beat is also a downbeat. The downbeat tracking algorithm is summarized in Figure 5.4. It consists of three main steps. First, features are extracted from the beat segments. Then, a logistic regression classifier, trained on 117 manually annotated songs, determines the probability that a beat is either the first, second, third or fourth beat in measure it belongs to. Finally, these predictions are aggregated over the entire song for each of the four options to determine the most likely downbeat positions.

For features, the loudness [31] of each beat fragment and the energy distribution of the audio along the frequency axis, binned in 12 equally spaced bins on the Mel frequency scale [30], are calculated. Additionally, three onset detection functions are calculated of the entire song, namely the *flux*, the *high-frequency coefficient* (HFC) and the *complex spectral difference* (CSD) ODF. Different ODFs capture different musical onset events [28] and informal experimentation indicated that using multiple ODFs greatly improves

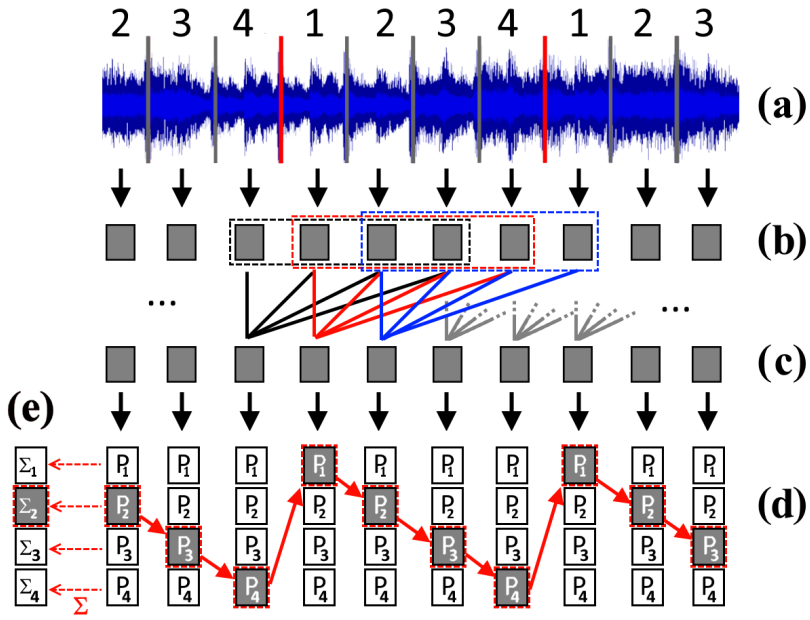


Figure 5.4: Overview of the downbeat tracking algorithm. (a) The audio waveform, annotated with beats and downbeats. (b) Isolated features are extracted from the beats. (c) Isolated features are combined to create contextual features. (d) The machine learning model classifies each beat, estimating the log-probability of a beat being either the 1^{st}, 2^{nd}, 3^{rd} or 4^{th} beat in a measure. (e) The downbeats are determined by summing the log-probabilities along each possible trajectory and choosing the most likely one. Here, the second trajectory is highlighted.

performance. With the notations from Table 5.2, this gives:

$$x_{\text{frame}}^{[N_{\hat{\tau}};N_{\hat{\tau}}]}(m) = x[N_{\hat{\tau}}(m) : N_{\hat{\tau}}(m+1)], \tag{5.12}$$

$$\mathcal{X}_{\text{loud}}(m) = \text{loudness}\left(x_{\text{frame}}^{[N_{\hat{\tau}};N_{\hat{\tau}}]}(m)\right), \tag{5.13}$$

$$\mathcal{X}_{\text{mel}}(m,k) = X_{\text{mel12}}\left(x_{\text{frame}}^{[N_{\hat{\tau}};N_{\hat{\tau}}]}(m)\right)(k), \tag{5.14}$$

$$\mathcal{X}_{\text{odf}_i}(m,k) = \Gamma_{\text{HWR}}^{(\text{i})}(L_{\text{beat}}^{(m)} + k),\ 0 \leq k < L_{\hat{\tau}}, i \in \{\text{flux}, \text{csd}, \text{hfc}\}. \tag{5.15}$$

The features are standardized by subtracting the mean and dividing by the standard deviation of the corresponding features in all beats of the song. These features describe each beat in isolation and are therefore called *isolated features*. However, a beat fragment does not contain enough information on its own to determine its position within its measure. Indeed, the notion of rhythmical structure arises by the carefully orchestrated accentuation of certain beats compared to other beats. Therefore, the proposed machine learning classifier uses so-called *contextual features* $\mathcal{X}^{\text{ctxt}}$ that describe *differences* between the isolated features of a given beat and those of the next 4 or 8 beats. For the different types of isolated features, they are calculated as follows, with m the current beat index, k the subfeature index if the isolated feature is a vector, l the lag in number of beats with which the isolated feature vector is compared, and $i \in \{\text{flux}, \text{csd}, \text{hfc}\}$:

$$\mathcal{X}_{\text{loud}}^{\text{ctxt}}(m,l) = \mathcal{X}_{\text{loud}}(m+l) - \mathcal{X}_{\text{loud}}(m), \qquad 0 < l < 8, \tag{5.16}$$

$$\mathcal{X}_{\text{mel}}^{\text{ctxt}}(m,l,k) = \mathcal{X}_{\text{mel}}(m+l,k) - \mathcal{X}_{\text{mel}}(m,k), \quad l \in \{-1,1,2,3\}, \tag{5.17}$$

$$\mathcal{X}_{\text{odf,corr}}^{\text{ctxt,(i)}}(m,l) = \sum_{k=0}^{L_{\hat{\tau}}-1} \Gamma^{(\text{i})}(L_{beat}^{(m+l)} + k)\Gamma^{(\text{i})}(L_{beat}^{(m)} + k), \quad l \in \{-1,1,2,3\}, \tag{5.18}$$

```
function get_trimmed_audio_start_and_end(audio x, beat period τ̂):
        X_rms(m) = sqrt( (1/N_τ̂) Σ_i { ( x_frame^[N_τ̂;N_τ̂](m,i) )^2 } ), ∀m ∈ {0,..,M−1}
        X̄_rms(m) = mean_{m−1≤q≤m+1} {X_rms(q)}
        X̄_max = max_m{X̄_rms(m)}
        for f in [0.9,0.8,...,0.1]:
            b_start = get_first_index_where(X̄_rms(m) > f*X̄_max)
            b_end = get_last_index_where(X̄_rms(m) > f*X̄_max)
            if b_end − b_start ≥ 0.4*M:
                return b_start, b_end
        return 0, M−1
```

Figure 5.5: Pseudo-code for the audio trimming algorithm of the downbeat tracker.

$$\mathcal{X}^{\text{ctxt,(i)}}_{\text{odf,int}}(m,l) = \sum_{k=0}^{L_{\hat{\tau}}-1} \Gamma^{(\text{i})}(L^{(m+l)}_{beat}+k) - \sum_{k=0}^{L_{\hat{\tau}}-1} \Gamma^{(\text{i})}((L^{(m)}_{beat}+k), \quad 0<l<16. \tag{5.19}$$

Initial experimentation indicated that the downbeat machine learning model is much less reliable in the less "expressive" parts of the audio, e.g. the intro and the outro. Therefore, beats in the intro and outro are trimmed away using a heuristic iterative algorithm based on the RMS energy of the beats. Figure 5.5 shows this algorithm in pseudo-code. First, the RMS value of every beat's audio is calculated. This sequence of RMS values is smoothed using a running mean operation, and then the first and last beat in the audio are determined where the running mean RMS value is larger than a threshold, i.e. a fraction of the maximum running mean RMS value. If more than 40% of all beats lie in between these boundaries, those beats are kept to determine the downbeats. Else, the threshold is decremented and the algorithm is repeated until at least 40% of the beats are 'selected'. The threshold fraction is initialized at 0.9 and is decremented in steps of 0.1.

To determine the downbeats of a song, the algorithm works as follows (see Figure 5.4). First, the audio is trimmed using the aforementioned algorithm. Then, the features of the remaining beats

are extracted and subsequently classified using the machine learning model. This results in a log-probability vector for each beat that estimates whether it is either the first, second, third or fourth beat in its measure. The algorithm then exploits the assumption that the music has a strict 4/4 time signature: for each of the four possible trajectories throughout the song, the corresponding log-probabilities of the beats are summed, and the trajectory with the highest sum is the most likely and is predicted to be correct. Finally, this prediction is extrapolated to the trimmed beats in the intro and outro, after which all downbeat positions are known.

Structural segmentation

The high-level compositional structure of the song is discovered using the novelty-based structural segmentation method by Foote [32]. This approach assumes that structural boundaries are accompanied by significant musical changes in the audio. It uses a so-called *structural similarity matrix* (SSM) $S(m, n)$, which generally is constructed by splitting the input audio into short frames, extracting features of those frames and comparing the features of each frame with those of every other frame, resulting in a matrix of pairwise comparisons. The automatic DJ system uses not one, but two SSMs, both using an analysis frame length of half a beat long ($N_{\hat{\tau}}/2$ samples) and a hop size of a quarter beat ($N_{\hat{\tau}}/4$ samples). As features, one SSM uses 13 MFCCs per measure, and the other uses the frames their RMS energy. The distance between the MFCC feature vectors is calculated using the cosine distance, for the RMS features the absolute scalar difference is used:

$$d_{\cos}(\vec{u}, \vec{v}) = 1 - \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|_2 \|\vec{v}\|_2}, \tag{5.20}$$

$$X_{\text{MFCC}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m, k) = X_{\text{MFCC}}\left(x_{\text{frame}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m)\right)(k), \tag{5.21}$$

$$S_{\text{MFCC}}(m, n) = d_{\cos}\left(X_{\text{MFCC}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m, \cdot), X_{\text{MFCC}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(n, \cdot)\right), \tag{5.22}$$

$$\mathcal{X}_{\text{RMS}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m) = \sqrt{\text{mean}\left\{\left(x_{\text{frame}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m)\right)^2\right\}}, \tag{5.23}$$

$$S_{\text{RMS}}(m,n) = \left|\mathcal{X}_{\text{RMS}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(m) - \mathcal{X}_{\text{RMS}}^{[N_{\hat{\tau}}/2;N_{\hat{\tau}}/4]}(n)\right|. \tag{5.24}$$

Similar frames lead to a low value in the matrix, whereas dissimilar frames result in high values (Figure 5.6a). The formation of blocks in the matrix indicates the occurrence of musically coherent segments. The algorithm by Foote determines the location of the boundaries between these segments by convolving the matrix along its main diagonal with a 'checkerboard' kernel $\mathcal{K}$:

$$\mathcal{K}(k,l) = \begin{cases} 1 & \text{for } kl \geq 0, \\ -1 & \text{for } kl < 0. \end{cases} \tag{5.25}$$

This leads to a *novelty curve* $\Gamma^{(\text{SSM,i})}(m)$, which has high values for time instants at a structural boundary and low values elsewhere (with $i \in \{\text{RMS}, \text{MFCC}\}$, and K the kernel width in frames):

$$\Gamma^{(\text{SSM,i})}(m) = \left|\sum_{k=-\frac{K}{2}}^{\frac{K}{2}} \sum_{l=-\frac{K}{2}}^{\frac{K}{2}} \mathcal{K}(k,l) S_i(m+k, m+l)\right|. \tag{5.26}$$

K is equal to 64 frames, i.e. 4 measures before and after each time instant are used to determine the novelty at that instant. This process is illustrated in Figure 5.6b. Finally, both novelty curves are combined by taking the element-wise geometric mean of both curves:

$$\Gamma^{(\text{SSM})}(m) = \sqrt{\Gamma^{(\text{SSM,RMS})}(m)\Gamma^{(\text{SSM,MFCC})}(m)}. \tag{5.27}$$

The geometric mean will only show a peak when both curves have a peak at the same time. Hence, the geometric mean gives the novelty curve for boundaries on which both individual curves 'agree'. This operation is based on the observation that boundaries between segments often go along with changes in

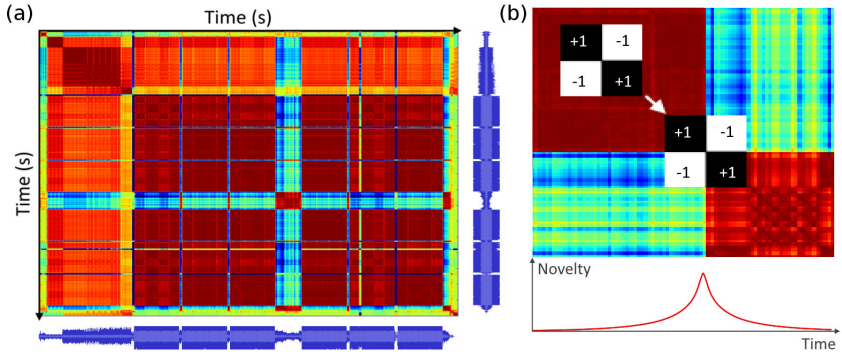


Figure 5.6: Illustration of structural segmentation concepts. (a) Example of a structural similarity matrix. The audio waveform is show next to both axes. (b) Calculation of the novelty curve by convolving the SSM with a checkerboard kernel along the diagonal.

harmonic content and musical texture as well as changes in musical energy [5, pp. 290–292, 297–298], which the MFCC SSM and RMS SSM respectively attempt to capture.

Given the audio's novelty curve, phrase-aligned structural segment boundaries are determined. Figure 5.7 summarizes this process in pseudo-code. Structural boundaries are extracted by first selecting the 20 highest peaks in the novelty curve. Several post-processing steps based on musically inspired rules are applied to determine the downbeat-aligned boundary positions. First, the peaks positions, which – given the SSM time resolution – are detected at a half-beat granularity and hence could initially be at non-downbeat positions, are rounded to the nearest downbeat. Peaks that lie further than 0.4 times the length of a measure from a downbeat are discarded as false positives. From the downbeat-aligned segment boundary candidates, a subset is determined in which all candidates lie at a multiple of 8 measures from each other, because phrases are assumed to be 8 measures long. In what follows, candidate boundary positions are denoted p_i (in

```
function get_boundaries(novelty_curve Γ^(SSM), beat period τ̂):
        p, a ← find_peaks_and_amplitudes(Γ^(SSM), 20)
        S, N ← initialize two arrays of length 8, initial values 0
        ignored ← initialize empty array
        for i in [0 .. 19]:
            if |p_i − ⌊p_i⌉| ≥ 0.4 * 4L^(SSM)_τ̂:
                ignored.append(i)
            else:
                p_i ← ⌊p_i⌉
        for i in [0 .. 19]:
            if i ∉ ignored and ∄j : (p_i < p_j ≤ p_i + 2) ∧ (a_j > 3/4 a_i):
                idx = mod(p_i, 8)
                S_idx = S_idx + a_i
                N_idx = N_idx + 1
        B_segment = argmax_B(N_B * S_B)
        return [p_i if mod(p_i, 8) == B_segment]
```

Figure 5.7: Pseudo-code for selecting phrase-aligned boundaries given the SSM novelty curve in the structural segmentation algorithm.

downbeats) and their amplitudes are denoted a_i. For each subset, all peak amplitudes are summed that do not occur one or two measures before another peak with a novelty value greater than 0.75 times the amplitude of the peak considered for summation. This reduces the number of false positives caused by *breaks*, i.e. short musical segments that *break up* the music and typically occur just before a structural boundary. Each sum is multiplied by the number of peaks contributing to it, increasing the importance of a sum with many contributors. The structural boundary candidates that contributed to the highest sum are considered to be the correct boundaries.

Each segment is then assigned a label based on its RMS energy: segments with an average RMS energy higher than the average song-wide RMS energy get a label 'high', whereas quieter segments get a label 'low'. This labeling method attempts to tag the *'main'* or *'core'* sections of the song with a label 'high' and the other sections (intro, buildup, breakdown, ...) as 'low'. This is used to determine cue points, as described in Sec. 5.3.2.

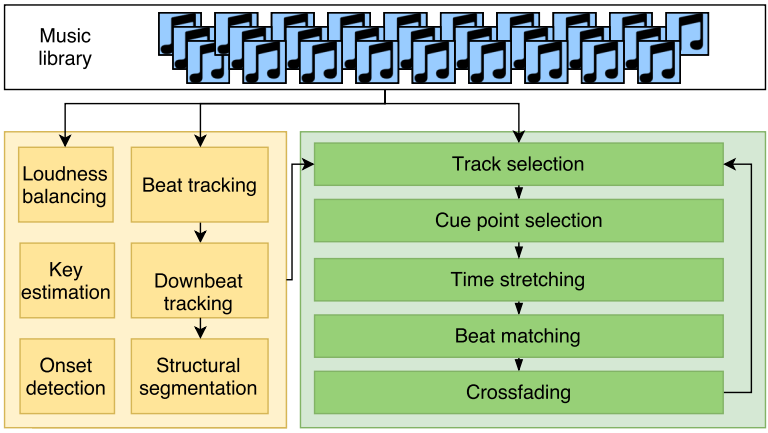


Figure 5.8: Automatic DJ system architecture. On the left, in yellow, are the annotation submodules. On the right, in green, are the modules used 'live' during the mixing process.

5.3.2 Composing a DJ mix

Section 5.3.1 described how the proposed DJ system analyzes music to gain a detailed understanding of it much like a human DJ has. The current Section elucidates how the proposed DJ system also simulates parts of the creative process of a human DJ, i.e. a creative track and cue point selection and a variation of transition types, leveraging the detailed musical understanding to generate high quality mixes.

Automatic DJ system architecture

The automatic DJ system architecture is shown in Figure 5.8. Songs are annotated offline using the beat tracker, downbeat tracker and structural segmentation modules. Additionally, the *replay gain* [33] is annotated, which allows to play all songs at an equal volume, regardless of the volume they were recorded at. Also the key (Sec. 5.3.2), *style descriptor* (Sec. 5.3.2) and the beat tracking ODF are determined. The mix generation and

playback happen 'live' by iteratively performing track and cue point selection, time-stretching, beatmatching and crossfading, as detailed below.

During the mixing process, the track and cue point selection algorithm determines the next song and cue points, given the current song. Three transition types, inspired by how professional DJs compose their mix and perform crossfades, are possible. The type is chosen using a Finite State Machine, which ensures variation in the mix by preventing that certain transition types happen twice in a row. The transition type defines which segments ('low' or 'high') of the first song and the new song are overlapped, i.e. it defines possible cue points (Sec. 5.3.2). The next song is selected such that the resulting mix is musically coherent as a whole while ensuring that individual song transitions are enjoyable to listen to. Details of this selection process are explained in Section 5.3.2.

Once the cue points are known, the crossfade is established. This happens by time-stretching the audio, applying volume fading and equalization filters and then adding the two songs together in a beatmatched way. The tempo is fixed at 175 BPM for the entire mix. This process is explained in more detail in Section 5.3.2.

Track and cue point selection

The track and cue point selection is done in five steps that iteratively narrow down the options for potential successors of the current song. Firstly, only songs that are *in key* with the current song are considered, i.e. songs in the same key or in a compatible key (see below). Of these songs, the six most similar songs in terms of musical style are selected as potential successors. In the third step, cue points are determined for the current song and for the candidate successors. The two final steps heuristically estimate the musical quality of each potential overlap. More specifically, vocal activity

is detected such that overlapping vocals of both songs is avoided, and rhythmic similarity is estimated by comparing the songs their ODFs. The song with the highest rhythmic similarity to the current song and without vocal clashes is selected as the next song.

Musical key compatibility Mixing in key or *harmonic mixing* is a technique often employed by professional DJs where successive songs have the same or a related key [3, 4]. This ensures that they fit together harmonically and reduces dissonance when playing two songs at the same time. Even though recent work [34] questions the use of harmonic mixing based on the musical key in specific cases, e.g. for music outside the major/minor scale framework, to the best of our knowledge, the technique is still common practice in drum'n'bass. Hence, the automatic DJ system features a simple key extraction implementation.

The key of each song is discovered in a pre-processing step using the algorithm by Faraldo et al. [35], which is implemented in the Essentia music analysis library [36] and has been tuned to electronic dance music in particular. The annotations are stored on disk, allowing fast retrieval. The allowed key changes for the next song are going a perfect fifth up or down (i.e. changing to the *dominant* or *subdominant* key) or changing to the *relative* key of the key of the current song. Only the songs that are in one of these keys or in the same key as the current song are considered in the following steps. This choice of allowed keys is inspired by DJing best practices [3, 13] and they are related to the music theory concept of the *circle of fifths*. This is illustrated in Figure 5.9 [26].

To give the automatic DJ more song options to choose from, also the songs in a key one semitone lower or higher than one of the related keys are added to the pool. If one of these would eventually be selected as the next song, its key will be altered using a process called *pitch shifting*. This involves subsequently time-stretching and then down- or upsampling the audio, effectively changing the

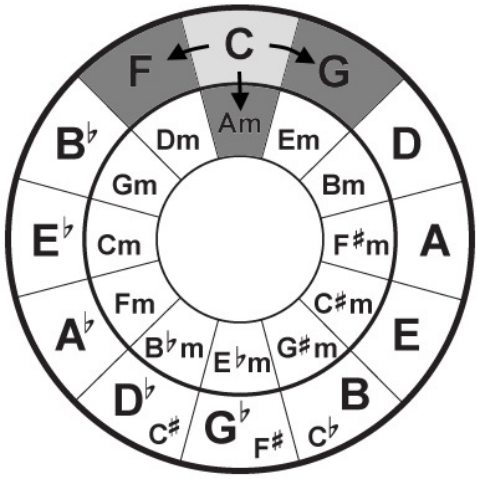


Figure 5.9: Illustration of the circle of fifths and allowed key changes in the DJ system. For example, if the key of the current song is C major (light gray), the allowed key changes are one perfect fifth up (G major, the *dominant* key), one perfect fifth down (F major, the *subdominant* key) or changing to the *relative key* (A minor).

music's pitch without altering the playback speed. This is necessary since a key one semitone below or above a compatible key is usually not a compatible key itself.

Musical style descriptor A DJ often mixes songs that are similar in *style*, *genre* or *atmosphere*. In what follows, the term *style* describes the combination of timbre, mood and energy of a song. For example, a song might be energetic or calm in nature, uplifting or moody, happy or dark and so on. The automatic DJ describes the style by means of a custom *style descriptor* feature.

Two much-cited approaches of content-based audio similarity measures are those by Logan [37] and by Aucouturier [38]. The former models the distribution of frame-wise spectral features by means of a K-means cluster model, whereas the latter constructs a Gaussian mixture model of the feature vectors. Song similarity is calculated using respectively the Earth Mover Distance algorithm and a sampling approach. However, these approaches are computationally expensive, both to construct the song models as for evaluating the distance metrics [39].

To limit the annotation and song comparison time, the automatic DJ adopts a simpler approach. First, *spectral contrast* features [40, 41] are extracted from short overlapping frames of audio, using a frame size and hop size of respectively 2048 and 1024 samples. Only the audio belonging to 'high' segments from the structural segmentation task are used, as these typically make up the most recognizable and descriptive parts of the music [5, p. 290]. The spectral contrast features calculate the ratio between the magnitudes of peaks and valleys within different sub-bands of the spectrum and in this way describe the relative amount of harmonic and noise components in those bands. They reportedly outperform MFCC features for music genre classification [41], as confirmed by informal experiments. Of these frame-wise features, the means and variances of the individual vector components are calculated across

all frames, and also the mean and variances of the first-order differences, resulting in a 48-dimensional vector describing each audio file:

$$\mathcal{X}_{\text{contrast}}^{[2048;1024]}(m,k) = \text{spectral-contrast}\left(x_{\text{frame}}^{[2048;1024]}(m)\right), \quad (5.28)$$

$$\mathcal{X}_{\text{style,mean}}(k) = \underset{m}{\text{mean}}\left(\mathcal{X}_{\text{contrast}}^{[2048;1024]}(m,k)\right), \quad (5.29)$$

$$\mathcal{X}_{\text{style,stdev}}(k) = \underset{m}{\text{stdev}}\left(\mathcal{X}_{\text{contrast}}^{[2048;1024]}(m,k)\right), \quad (5.30)$$

$$\mathcal{X}_{\text{style},\Delta,\text{mean}}(k) = \underset{m}{\text{mean}}\Big(\mathcal{X}_{\text{contrast}}^{[2048;1024]}(m,k) - \mathcal{X}_{\text{contrast}}^{[2048;1024]}(m-1,k)\Big), \quad (5.31)$$

$$\mathcal{X}_{\text{style},\Delta,\text{stdev}}(k) = \underset{m}{\text{stdev}}\Big(\mathcal{X}_{\text{contrast}}^{[2048;1024]}(m,k) - \mathcal{X}_{\text{contrast}}^{[2048;1024]}(m-1,k)\Big), \quad (5.32)$$

$$\mathcal{X}_{\text{style}} = [\mathcal{X}_{\text{style,mean}}, \mathcal{X}_{\text{style,stdev}}, \mathcal{X}_{\text{style},\Delta,\text{mean}}, \mathcal{X}_{\text{style},\Delta,\text{stdev}}]. \quad (5.33)$$

The vector $\mathcal{X}_{\text{style}}$ is then projected onto a lower-dimensional feature space in which thematically similar songs lie close together and dissimilar songs are further apart. This projection is done by projecting them onto the top three dimensions found by applying Principal Component Analysis (PCA) on a collection of 160 drum'n'bass songs. The resulting three-dimensional representation will be called the *style descriptor* $\Theta = \text{PCA}_3(\mathcal{X}_{\text{style}})$. The Euclidean distance between two style descriptors measures song similarity.

Before describing how the DJ system's track selection algorithm uses the style descriptor, an analysis of the behavior of the style descriptor in professional DJ mixes is presented to motivate our approach. In this analysis, several professional mixes downloaded from the internet were analyzed by extracting 90 seconds of music with a hop size of 120 seconds. These parameters were chosen

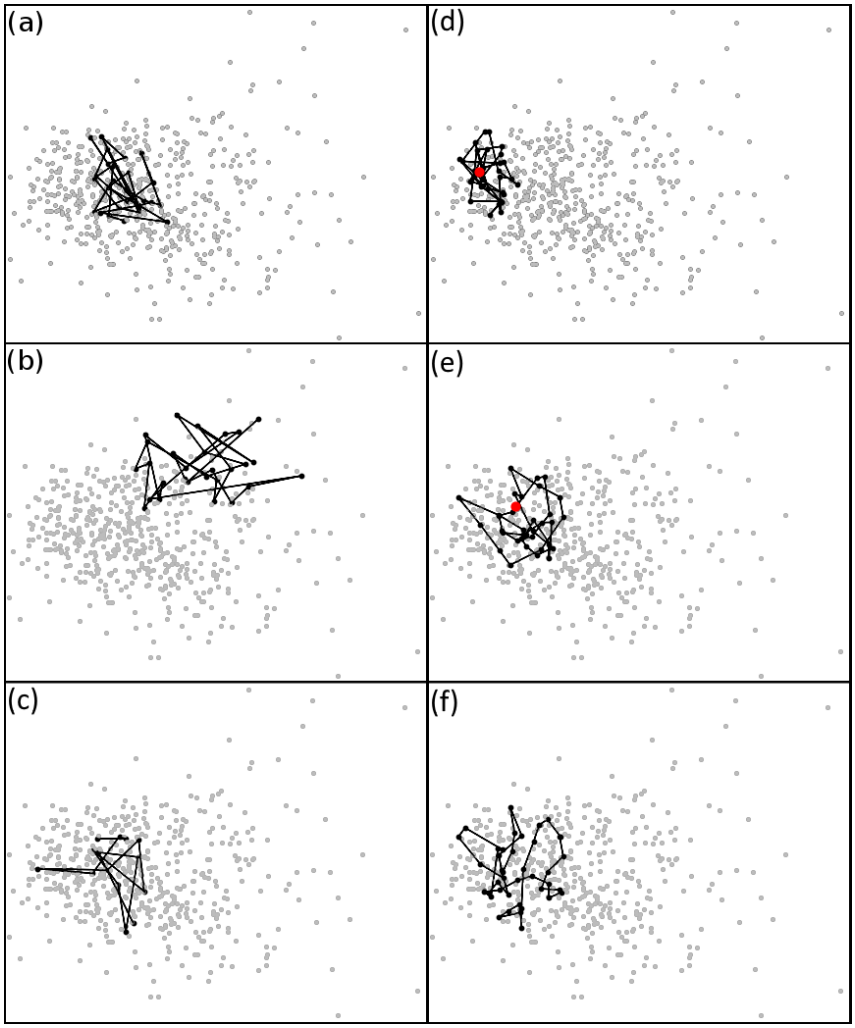


Figure 5.10: Illustration of the style descriptor progression throughout six DJ mixes. The gray dots represent the first two PCA components of the style descriptors of 513 drum'n'bass songs, selected from different subgenres of drum'n'bass. The black dots mark the songs in the mix. Successive songs are connected by a line. (a,b,c) Professional DJ mixes, resp. [42], [43], [44]. (d,e,f) Generated DJ mixes. The style centroid is marked by a red dot. (d) Default settings for $(\alpha, \beta) = (0.4, -0.1)$, (e) Large β: $(\alpha, \beta) = (0.4, -0.9)$, (f) No style centroid, large β: $(\alpha, \beta) = (0, -0.5)$.

after informally determining the average duration of an individual song in a DJ mix. The evolution of the style descriptor throughout three professional DJ mixes is visualized in Figure 5.10(a-c). The style stays in a relatively small neighborhood compared to the entire collection of songs in the music library. Therefore, to imitate this behavior, the automatic DJ system should also select songs that stay relatively close to one another (Figure 5.10(d)). Note that this observation also strengthens the hypothesis that the style descriptor groups thematically similar songs close together, since professional DJs typically select quite similar songs for a mix [5, pp. 317–320].

The song selection process works as follows. Before the mix is started, the automatic DJ randomly selects the first song and selects the quarter of all the songs in the music library that are closest to this first song in the style descriptor space. The centroid of the style descriptors of these songs will be called the *style centroid* $\Theta_{centroid}$. During generation, the DJ calculates the ideal style of the next song $\hat{\Theta}_{next}$ as a weighted sum of the current song's descriptor, the previous song's descriptor, and the style centroid. With α and β the weight of the style centroid and of the previous song respectively, this gives:

$$\hat{\Theta}_{next} = \alpha\,\Theta_{centroid} + (1-\alpha)\,(\beta\,\Theta_{prev} + (1-\beta)\,\Theta_{cur})\,. \qquad (5.34)$$

α and β have been arbitrarily assigned the values 0.4 and -0.1 respectively. The centroid weight α ensures that all songs stay relatively close to the style centroid, leading to a thematically uniform mix. A value of α close to 1 will keep the mix centered around the style centroid, while a value close to 0 allows the mix to "wander" freely through the style space, leading to a more varied but potentially less uniform mix (Figure 5.10(e)). A negative weight for β is chosen such that there is a deliberate progression or 'flow' of the style: the next song will be dissimilar to the current song in a similar way as the current song is dissimilar to the previous song. A more negative β leads to larger the jumps in

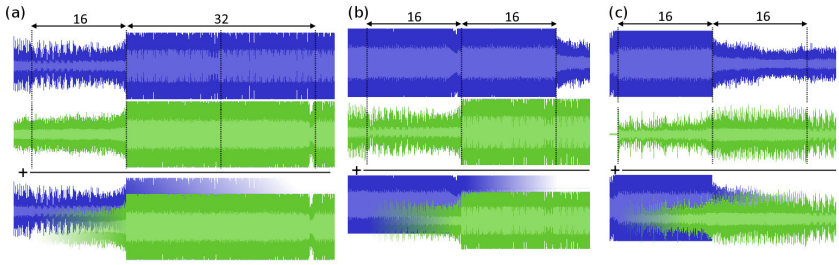


Figure 5.11: Illustration of the cue point selection for the three transition types. The fade-in and fade-out lengths, expressed in number of measures, are annotated. Also the final crossfades are conceptually illustrated. **(a)** Double drop. **(b)** Rolling transition. **(c)** Relaxed transition.

style between successive songs (Figure 5.10(f)), again presenting a trade-off between variation and coherence. Figure 5.10(d) illustrates the default behavior of the automatic DJ system, while Figure 5.10(e) and Figure 5.10(f) show the effect of alternative settings for α and β.

The pool of song candidates is reduced by only keeping the songs that are closest to $\hat{\Theta}_{next}$. In the current prototype of the automatic DJ, six songs are kept.

Cue point selection To determine cue points, the automatic DJ first chooses one out of three transition types using a pseudo-random selection process that is described below. These types differ in what types of structural segments are overlapped ('*high*' or '*low*' energy) and hence determine the allowed cue point positions. For the different transition types, this selection happens as follows (as illustrated in Figure 5.11):

- *Double drop*: in both the current and the next song, the cue points are chosen 16 measures before a drop, i.e. a transition from a low to a high segment. In the current song, the first

drop after the current playback instant is used; in the next song, if there are multiple drops, then one is picked at random. The fade-in and fade-out are 16 and 32 measures long respectively. The drops of both songs are aligned in time, hence the name *double drop*.

- *Rolling transition*: the cue point in the current song is 32 measures before the next transition from a high to a low segment. The cue point in the next song is the downbeat 16 measures before a drop. The fade-in and fade-out are both 16 measures long. In this transition, the *drop* of the second song continues the energetic nature of the high section of the preceding song.

- *Relaxed transition*: the cue point in the current song is 16 measures before the next transition from a high to a low segment. The next song is started at the beginning, i.e. its first structural boundary. The fade-in and fade-out are both 16 measures long. In this way, the energetic nature of the mix is interrupted, giving the listener some time to relax.

The transition type is chosen pseudo-randomly by means of a finite state machine that specifies the probability of selecting each type, given the previous transition's type (Table 5.3). The current implementation balances a rather high-energy 'flow' (high probabilities for rolling transitions) with moments to rest (relaxed transitions) and exciting climaxes (double drops). To avoid layering too many songs at once, double drops are not allowed after a rolling transition or another double drop.

After determining the transition type, cue points are determined for each of the six remaining song candidates. Given that particular choice of cue points, the automatic DJ system estimates the quality of the crossfades in the two final steps to make a final selection for the next song.

Table 5.3: Transition matrix controlling the transition type selection. The element in the ith row and jth column denotes the probability of selecting type j if the previous transition's type is i.

	Relaxed	*Rolling*	*Double drop*
Relaxed	0	0.7	0.3
Rolling	0.2	0.8	0
Double drop	0.2	0.8	0

Vocal clash detection When two songs are mixed together, it is important to avoid vocal clashes as this might disrupt the listening experience, especially when the vocals have a predominant presence in both songs [5, p. 318]. Even though drum'n'bass is a predominantly instrumental genre, vocals regularly occur. For this reason, a vocal activity detection mechanism is built into the automatic DJ system.

Existing work on singing voice detection [45–47] typically adopts a machine learning approach, where features extracted from short frames of audio are classified as either vocal or instrumental. The output is then smoothed using e.g. an HMM or ARMA filtering [47, 48]. Unfortunately, at the time of implementing the DJ system, no software implementation of a vocal activity detection solution was available that could be easily integrated into the automatic DJ system. A simple baseline solution is developed instead. A support vector machine classifier with an RBF kernel is trained to classify segments of one measure long. The segments are first analyzed in frames of 2048 samples long with a halve frame overlap between consecutive frames. Of these short frames, 13 MFCCs, 6 spectral contrast and the corresponding 6 spectral valley features [41] are calculated. The means, variances and skews of these features are calculated, as well as the means and variances of the first order differences of features in consecutive frames, giving 250 features for each downbeat segment. These are the input to the SVM classifier. The training set consists out of 55

manually annotated songs. The classifier parameters were tuned using cross-validation, giving a regularization parameter C of 1.0 and kernel coefficient γ of 0.01. During the annotation process, the SVM classification of each measure is stored on disk to be used in the song selection process.

The classification output is post-processed before using it for song selection. A measure is labeled 'vocal' if either the measure itself or both its neighboring measures are classified as 'vocal' by the SVM: else the measure gets the label 'instrumental'. A vocal clash is detected if at least two 'vocal' measures overlap; a one-measure overlap is tolerated. Only the songs and cue points are considered that do not lead to a vocal clash. If however all candidates lead to a clash, the clash is tolerated for that transition.

Onset detection function matching Rhythmic similarity is estimated by comparing the songs their onset detection functions. The ODFs from the beat tracking process are stored on disk and are used for this comparison. The ODF portions corresponding to the overlapped music segments are compared by means of the dynamic time warping (DTW) algorithm, since the ODFs are calculated for the audio at their original (unstretched) tempo. The DTW algorithm stretches the ODFs during comparison and hence eliminates the need to recalculate the ODFs after time-stretching the original audio. A second motivation for the DTW algorithm is that small rounding errors occur when selecting an extract from the ODFs, as beat positions often do not align with an ODF frame boundary. Because of this, two ODF extracts that are very alike might be shifted one or two frames with respect to each other: therefore, some flexibility is allowed. However, a beam search strategy is applied to prevent excessive warping of the compared ODFs. The song and cue point combination which leads to the lowest DTW score is selected as optimal.

Creating the crossfade

Given the next song and the cue point positions, the crossfade is established. Time-stretching is performed by applying harmonic-percussive separation [49] and independently stretching the harmonic and percussive residuals, respectively using the frequency-domain phase vocoder [50] and the time-domain OLA time scale modification algorithm [51], as proposed by Driedger and Müller [52]. This approach significantly reduces audio artifacts compared to traditional time-stretching algorithms at the expense of a higher computational cost [53]. Afterwards, volume fading and equalization filters are applied to both songs. The exact crossfade patterns for each transition type were manually defined to ensure a smooth transition between songs. During the fade-in, the volume of the next song gradually rises to full level, while its bass and treble remain almost entirely filtered out. At the end of the fade-in, a *switch* occurs between the songs: the bass and treble of the next song are quickly increased to full level, and at the same time the bass and treble of the previous song are filtered out. The volume of the previous song is then gradually faded out until the end of the crossfade. This particular equalization pattern ensures that the bass and treble of only one song will be heard at any point during the crossfade, leading to a clean, unsaturated sound of the resulting mix. Finally, the audio of the songs is added together in a beatmatched fashion, effectively mixing the two songs together.

5.4 Results and Discussion

The automatic DJ system has been tested on different aspects. First, the system's execution time is measured and the envisaged setup is described. Then, the accuracy of the different annotation methods from Section 5.3.1 is evaluated. Additionally, the vocal activity detection algorithm is evaluated on its ability to discriminate between clashing and non-clashing crossfades. Finally, a subjective user test assesses the efficacy of the style descriptor and the ODF

matching and tests the system as a whole.

5.4.1 System performance and envisaged setup

The presented DJ system mixes "live", i.e. it generates future transitions in the background while the music is playing. On a consumer-grade laptop (4-thread Intel Core i7-6500U CPU at 2.50GHz, 8GB RAM), it takes 17.0s on average to generate the next transition (averaged over 100 generated transitions; standard deviation 6.6s, maximum 32.2s). This is less than 16 measures of music at 175 BPM ($\approx$ 22s). In the current implementation, two consecutive transitions start at least 16 measures (and typically more) apart from each other, ensuring that the system doesn't hang because it's waiting for the next transition to be generated. Furthermore, a buffer mechanism queues multiple transitions as an additional safeguard that prevents the system from running out of music.

The annotation of the input music library happens offline, i.e. before the actual mixing process can start. Annotating one song takes 12.0s on average, i.e. 2.3s per minute of audio on average (standard deviation 0.1s; evaluated on 160 songs). As a coarse guideline, we recommend at least 50 to 100 drum'n'bass songs in the input music library in order to create a 30 minute mix ($\approx$ 20 transitions) of good quality. Of course, the more songs in the library, the more flexibility the automatic DJ has, and the better the mix will be.

5.4.2 Evaluation of the beat tracker

The beat tracking accuracy is evaluated on a collection of 160 drum'n'bass songs (see Appendix B). The annotations made by the beat tracker are not compared with a manually annotated ground truth, because this manual labeling is extremely time consuming and because the inherent ambiguity of the beat annotation task [54] makes this even more difficult. Instead, the annotation correctness

is evaluated by overlaying the audio with beep sounds at the beat positions detected by the algorithm and listening if they coincide with the true beat positions in the music. To avoid listening to the entire song, two 15-second fragments are evaluated per song: one a quarter into the song, and one three quarters into the song. If a fragments is beatless (e.g. because at that moment in the song there happens to be a breakdown), a new fragment with beats is chosen near that position and is evaluated instead. Large beat tracking errors, such as a large tempo error (e.g. 1 BPM or more) or a wrong phase, are clearly audible when listening to only one of the fragments. The second fragment is used to detect small tempo errors: even if in the first fragment the beat positions appear to be correct, they are unlikely to be correct in the second fragment because the tempo deviation would cause the beat annotations to drift from the correct beat positions. Among different candidates, the *melflux* ODF, as implemented in the Essentia library [36], performs best with 159 out of 160 songs annotated correctly, i.e. an accuracy of 99.4%. A second evaluation on a second test set of 220 songs leads to an accuracy of 98.2%, i.e. 4 songs their beats are incorrectly detected. In these cases, the tempo is still detected correctly, but the phase is wrong by half a beat period, such that the detected beat positions lie exactly in between the correct beats.

To motivate the need for a custom beat tracker, this paragraph discusses the behavior of existing beat tracker implementations. Experiments with several open-source implementations [36, 55] show that these annotate beat positions that are often not exactly equidistant, either because the algorithms are tailored towards songs with a variable tempo or because the specific implementation is not entirely accurate. We illustrate this in more detail for the `BeatDetectionProcessor` beat tracker from the *madmom* music processing library [55] on the first and second set of songs (resp. 160 and 220 songs). Even though the *madmom* documentation claims it detects beats at a constant tempo, slight inter-beat interval (IBI) variations occur: on average, the IBI standard deviation within a

song is 2.2% of the correct IBI length. This happens because it detects the IBI as an *integer* number of ODF frames, even though the IBI usually is a non-integer number of frames long, and then fixes rounding errors by detecting beats using a tolerance interval. Using these non-equidistant beat positions would complicate e.g. the beat matching implementation, and a constant IBI throughout the entire song is much more desirable. However, averaging the IBIs leads to significant tempo estimation errors ($\hat{\nu} = \frac{60}{mean_i(IBI_i)}$): for 115 out of 380 tested songs (30.3%), the estimation differs more than 0.25 BPM from the real tempo, and for 59 songs (15.5%), the error is even higher than 0.5 BPM. This is considered too inaccurate for an automatic DJ system, explaining the need for a custom beat tracker that uses an exact, constant IBI between all beats.

5.4.3 Evaluation of the downbeat tracker

The downbeat tracking machine learning model has been trained on a training set of 117 annotated songs, and its performance is evaluated on a hold-out test set of 43 songs. The ground-truth downbeat position is annotated as a single label per song (*1*,*2*,*3* or *4*), which denotes whether the first beat of the song, and every fourth beat after that, is either the 1^{st}, 2^{nd}, 3^{rd} or 4^{th} beat in the measure it belongs to. This of course assumes that the beat positions are annotated: these annotations are created using the algorithm described in section 5.3.1 and manually determining the tempo and phase of the songs where the beat tracker made a mistake. On individual beat segments, i.e. by taking the argmax for each per-beat log-probability prediction (step (d) in Figure 5.4) and before summing the log-probabilities (so before step (e) in Figure 5.4), the downbeat classifier achieves an accuracy of 75.4% (excluding beats in the intro and outro). On a song level, i.e after summing the log-probability vectors of the individual beats and predicting a single song-wide downbeat index, the downbeat tracker achieves an accuracy of 95.3% on the 43 test songs. On the second test set of 220 songs, four of the 214 songs with correct

beat annotations had incorrect downbeat annotations, leading to an accuracy of 98.1%.

The need for a custom downbeat tracker is motivated mainly because existing downbeat trackers implemented in open-source libraries were found to be less accurate on this corpus of music and additionally require more computation time. We evaluated this in more detail by running the `RNNBarProcessor` downbeat tracker from the madmom library [55] on 160 songs, given the ground-truth beat annotations as input. The algorithm takes 35.4 seconds per song on average (standard deviation 6.7s) to calculate the downbeat positions, compared to 4.47 seconds (standard deviation 1.4s) for the DJ system's implementation. Furthermore, 20 songs their downbeat position were wrong, which gives an accuracy that is considerably worse than the presented downbeat tracker (87.5% compared to 98.1%).

5.4.4 Evaluation of the structural segmentation task

The accuracy of the structural segmentation task is more difficult to assess because of its subjective nature [56]. Here, annotations are considered *structurally correct* if they belong to the correct subset of downbeats at a multiple of eight measures from each other (see Sec. 5.3.1). Additionally, the simple labeling of the segments as a 'low' or a 'high' segment is evaluated. To do so, every transition from a 'low' to a 'high' segment in each of the 160 songs is listened to and is considered correct if that transition coincides with a *drop* in the music (see Figure 5.2). For 3 out of 160 songs, the 8-measure offset was incorrect. For 4 out of 160 songs, the assumption that all segmentation boundaries lie at a multiple of 8 measures from each other is incorrect. This means that 95.3% of the songs received structurally correct annotations. For 82.4% of these songs, also the drop detection as a 'low' to a 'high' transition is correct. In the second test set, 94.3% of the songs with correct downbeat annotations also received structurally correct segmentation annotations.

Table 5.4: Confusion matrix for the vocal clash detection experiment (No clash: 10167 crossfades; Clash: 1524 crossfades)

	No clash (estimated)	Clash (estimated)
No clash (Ground truth)	0.835	0.165
Clash (Ground truth)	0.307	0.693

5.4.5 Evaluation of the song filtering

The SVM classifier for vocal activity detection has been trained on a manually annotated selection of 55 songs, containing 12260 downbeat segments in total of which 2321 contain vocal activity. Data augmentation is performed by pitch shifting the audio for training one semitone up and down, artificially tripling the amount of training data. The unequal class sizes are countered by choosing class weights inversely proportional to the class size [57]. The classifier achieves an accuracy, precision, recall and F1-score of respectively 97.8%, 90.2%, 99.2% and 94.5% on the train set and 87.0%, 62.3%, 61.1% and 61.7% on a hold-out test set of 38 manually annotated songs containing 8524 downbeats (7097 non-vocal and 1427 vocal downbeats), where the *vocal* label is considered to be the positive class.

The ability to detect vocal clashes in transitions is evaluated by calculating all overlaps between all pairs of manually annotated songs that would be possible in the automatic DJ system. Each overlap is assigned a label *clashing* if vocals overlap in at least two measures in the resulting mix (as in the DJ system, an overlap of one measure is tolerated). The same label is estimated using the singing voice detection model. The confusion matrix for this experiment is shown in Table 5.4. This shows that about 69.3% of the vocal clashes can be prevented, at the expense of falsely discarding 16.5% of the non-clashing crossfades.

Table 5.5: User test results

	Mean score	Stdev score	Favorited
(a) standard autoDJ	3.89	0.90	94.4% (17)
(a) style-agnostic autoDJ	2.39	0.92	5.6% (1)
(b) standard autoDJ	3.76	0.90	66.7% (10)
(b) inverse ODF matching	3.41	1.12	33.3% (5)

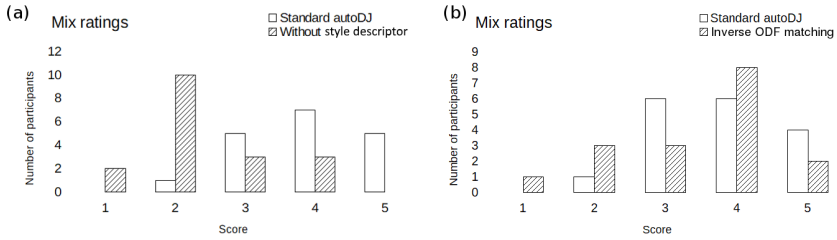


Figure 5.12: User test results: comparison of the ratings by the users. (a) Style descriptor track selection vs. style-agnostic track selection (b) Automatic DJ vs. song selection of worst matching songs in terms of ODF overlap

5.4.6 User study: Evaluation of the musical style descriptor and system evaluation

Figure 5.10 informally validates the ability of the style descriptor feature to group similar songs close together and separate dissimilar songs. To provide a more robust evaluation, a subjective user test has been carried out.

Two mixes are generated using an input library of 380 songs. One mix has all automatic DJ functionality enabled, in the other the songs are selected without considering the style descriptor. The mixes are both 15 minutes long and contain 10 and 11 transitions respectively. Both only use songs with structurally

correct annotations, to ensure that any loss in mix quality is the result of the track selection process and not of an annotation error. Additionally, both mixes start with the same song. 18 drum'n'bass fans were asked to listen to both mixes and select which mix had the best *track selection* in their opinion. These fans were reached through online drum'n'bass fora; they were not paid to take part in the experiment. The users are told that both the mixes are generated by the DJ system. They also rated each mix with a score from 1 to 5, where 5 is the best possible score. Histograms of these ratings are shown in Figure 5.12. In the same user test, two other mixes are presented to the users using a similar procedure to evaluate the ODF matching procedure. In the first mix, all automatic DJ functionality is enabled, and in the second the ODF matching is reversed such that the *worst* matching song is selected instead of the best match. The ratings for both mixes are compared in Figure 5.12. The user test results are shown in Table 5.5.

This test indicates that the style descriptor selection process drastically improves the mix consistency and its perceived quality, with 17 out of 18 evaluators preferring that mix (p-value 7.25×10^{-5}). Figure 5.12(a) also shows that the mix with the style descriptor received higher ratings on average (an average rating of 3.89 versus 2.39 respectively). The ODF matching mix was also preferred more often than the reversed matching scenario, but the distinction is less clear (10 out of 15 evaluators[2], p-value 0.151, and an average rating of 3.76 versus 3.41 respectively). The user tests also resulted in valuable qualitative feedback[3]. The first mix with all DJ features was received very well. Compared to the second mix (style-agnostic track selection), it was said to be *"more consistent"*, *"tunes that match in style and feel"*, that it had *"more continuity"* and less dissonance. The mix with style-agnostic track

[2]One evaluator only filled out half the user test, and two could not decide which mix sounds better, giving three evaluations less for the ODF matching.

[3]The user test results are published online, see [58].

selection was critiqued as being *"too rushed"*, changing too much between subgenres and that it *"sounded like a DJ that was occasionally just throwing tunes into the mix without much thought"*. Only one user preferred the second mix over the first, because *"it incorporates more vocals, this makes me think that it is more likely to be made by a human"*. The transition quality for the second set of mixes was more or less the same according to many participants, which indicates that selecting songs based on ODF similarity might not significantly influence the mix quality. Nevertheless, as there is a slight preference towards the ODF-based selection, this feature is kept in the DJ system. Other informal feedback indicates that the key detection algorithm and the vocal activity detection might need improvement, as some users noticed bad harmonic mixing or vocal clashes in some transitions.

5.5 Conclusions

The automatic DJ system presented in this chapter generates a seamless music mix from a library of drum'n'bass songs. It uses beat tracking, downbeat tracking and structural segmentation algorithms to analyze the structural properties of the music. Evaluated on a hold-out test set of 220 songs, a beat tracking accuracy of 98.2%, a downbeat tracking accuracy of 98.1% and a segmentation accuracy of 94.3% is achieved, meaning that in total 90.9% of the songs get structurally correct annotations. This efficacy is made possible in part by adopting a genre-specific approach. The extracted structural knowledge is used in a rule-based framework, inspired by DJing best practices, to create transitions between songs. To the best of the authors' knowledge, this system is the first that combines all basic DJing best practices in one integrated solution. Musical consistency between songs is ensured by employing a harmonic mixing technique and a custom *style descriptor* feature. The track selection algorithm also optimizes the rhythmic onset pattern similarity in the overlapped music segments and avoids vocal clashes by means of a support vector

machine classifier. A subjective evaluation in which 18 drum'n'bass fans participated indicates that the style descriptor feature for song selection drastically improves the mix quality. The influence of the ODF similarity matching is less clear. Overall, the automatic DJ system is able to seamlessly join together drum'n'bass songs and create an enjoyable mix.

However, some aspects of the system could still be evaluated in more detail, while others could be improved or extended. An obvious direction of further evaluation is to adapt the proposed system to different EDM genres. For some genres, this can be as straightforward as adapting the tempo range, and potentially retraining the downbeat tracker on some annotated songs in the target genre. However, as different genres might require different styles of mixing [4], for some genres, more modifications might be required (e.g. implementing additional transition types). Additionally, comparing the different transition types or even the DJ system as a whole with e.g. transitions performed by a human expert or a commercial automatic DJ system would also be very informative, and offers a great direction of further research. Several extensions and augmentations are also possible. Firstly, the annotation modules would benefit from a built-in reliability measure to discard songs with uncertain annotations. Furthermore, the used vocal activity detection and key estimation algorithms are simple baseline versions and subject to improvement. To improve the cue point selection, a finer distinction between section types could be implemented, instead of only considering 'low' and 'high' sections. This could lead to more precise cue point positions or a more diverse set of transition types. It would furthermore be very interesting to make the volume and equalization progressions throughout the transitions adapt to the songs being mixed, instead of following a fixed pattern as in the current implementation. Finally, even though the track selection method yields pleasing results, it might be an oversimplification of how professional DJs compose their sets and create a deliberate progression of energy

and mood throughout the mix. A model that better understands the long-term structures and composition of a good drum'n'bass mix, e.g. learned using the abundance of professional drum'n'bass mixes that can be found online, might therefore lead to a better result.

To conclude, we highlight the potential of an automatic DJing system, e.g. integration in audio streaming services to create an infinite DJ mix from a huge collection of music, as an assisting tool for human DJs to quickly explore their music libraries, or even as a substitute for a human DJ. Allowing the human DJ or the audience to interact with the automatic DJ system is also a very intriguing direction of future work, as this could lead to many new creative applications that were previously not possible. This automated approach is furthermore not constrained by the limitations of a human: it can mix together an arbitrary number of songs with fine-grained control over the equalizer settings, explore a myriad of song combinations before choosing the optimal one, and modify or even generate music on the fly. It remains an open question whether a computer could eventually become equally good or even better than professional DJs. After all, what characterizes good DJs is a certain performance element, their strong connection with the crowd, their deep insight in the music and their ability to consistently surprise in so many creative ways—aspects that are challenging to automate. Nevertheless, automated DJ systems have a great creative potential, and open up many interesting options for both listeners and DJs.

References

[1] Len Vande Veire and Tijl De Bie. "From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass." In: *EURASIP Journal on Audio, Speech and Music Processing* 2018.13 (2018). DOI: https://doi.org/10.1186/s13636-018-0134-8.

[2] Len Vande Veire. *From raw audio to a seamless mix: an Artificial Intelligence approach to creating an automated DJ system.* Ghent University, 2017.

[3] John Steventon. *DJing for dummies.* Hoboken, NJ: John Wiley & Sons, 2014. ISBN: 978047003275.

[4] Frank Broughton and Bill Brewster. *How to DJ (properly) - The art and science of playing records.* London, UK: Bantam Press, 2006. ISBN: 9780593049662.

[5] Mark Jonathan Butler. *Unlocking the groove: Rhythm, meter, and musical design in electronic dance music.* Bloomington, IN: Indiana University Press, 2006.

[6] Tristan Jehan. "Creating music by listening". PhD thesis. Massachusetts Institute of Technology, 2005.

[7] David Bouckenhove. *Automatisch Mixen Van Muzieknummers Op Basis Van Tempo, Zang, Energie En Akkoordinformatie.* Ghent University, 2007.

[8] Heng-Yi Lin, Yin-Tzu Lin, Ming-Chun Tien, and Ja-Ling Wu. "Music Paste: Concatenating Music Clips Based on Chroma and Rhythm Features". In: *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009), Kobe, Japan* (2009), pp. 213–218.

[9] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. "Full-Automatic DJ Mixing System with Optimal Tempo Adjustment based on Measurement Function of User Discomfort." In: *Proceedings of the 10th International Society*

for Music Information Retrieval Conference (ISMIR 2009), Kobe, Japan (2009), pp. 135–140.

[10] Tatsunori Hirai. "MusicMixer : Computer-Aided DJ System based on an Automatic Song Mixing". In: *Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology*. 2015. ISBN: 9781450338523.

[11] Matthew E P Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. "AutoMashUpper: Automatic creation of multi-song music mashups". In: *IEEE/ACM Transactions on Speech and Language Processing (TASLP)* 22.12 (2014), pp. 1726–1737. ISSN: 23299290. DOI: 10.1109/TASLP.2014.2347135.

[12] Chuan-lung Lee, Yin-tzu Lin, Zun-ren Yao, and Feng-yi Lee. "Automatic Mashup Creation by Considering both Vertical and Horizontal Mashabilities". In: *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015), Málaga, Spain* (2015), pp. 399–405.

[13] Mixed In Key LLC. *Mixed In Key software*. 2018. URL: https://mixedinkey.com/ (visited on 07/06/2018).

[14] Mixed In Key LLC. *Mashup2 software*. 2018. URL: https://mixedinkey.com/ (visited on 07/06/2018).

[15] Serato. *Serato DJ software*. 2017. URL: https://serato.com/dj (visited on 05/15/2017).

[16] Native Instruments GmbH. *Traktor Pro 2*. 2017. URL: https://www.native-instruments.com/en/products/traktor/dj-software/ (visited on 05/15/2017).

[17] Atomix Productions. *Virtual DJ*. 2017. URL: https://www.virtualdj.com/ (visited on 05/15/2017).

[18] Mixxx. *Mixxx*. 2017. URL: https://www.mixxx.org/ (visited on 05/15/2017).

[19] Serato. *Serato Pyro*. 2017. URL: https://seratopyro.com/ (visited on 05/15/2017).

[20] Pacemaker Music AB. *Pacemaker*. 2017. URL: https://pacemaker.net/ (visited on 05/15/2017).

[21] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. "AutoMashUpper: An Automatic Multi-Song Mashup System." In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013), Curitiba, Brazil* (2013), pp. 575–580.

[22] Tue Haste Andersen. "In the Mixxx: Novel Digital DJ Interfaces". In: *CHI 2005, Portland, Oregon, USA* (2005), pp. 1136–1137. DOI: 10.1145/1056808.1056850.

[23] Dan White. *Casual DJ Mixing Apps in Pacemaker + Spotify: Should DJs Be Worried?* 2015. URL: http://djtechtools.com/2015/12/17/casual-dj-mixing-app-spotify-and-pacemaker/ (visited on 09/05/2017).

[24] Dan White. *Serato Pyro: Can A Casual App Take Over House Party DJing?* 2016. URL: http://djtechtools.com/2016/02/11/serato-pyro-can-a-casual-app-take-over-house-party-djing/ (visited on 09/05/2017).

[25] Jason Hockman, Matthew E P Davies, and Ichiro Fujinaga. "One in the Jungle: Downbeat Detection in Hardcore, Jungle, and Drum and Bass." In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012), Porto, Portugal* (2012), pp. 169–174.

[26] Michael Pilhofer and Holly Day. *Music theory for dummies*. Hoboken, NJ: John Wiley & Sons, 2015.

[27] Matthew E P Davies and Mark D. Plumbley. "Context-dependent beat tracking of musical audio". In: *IEEE Transactions on Audio, Speech and Language Processing* 15.3 (2007), pp. 1009–1020. ISSN: 15587916. DOI: 10.1109/TASL.2006.885257.

[28] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. "A Tutorial on Onset Detection in Music Signals". In: *IEEE Transactions on Speech and Audio Processing* 13.5 (2005), pp. 1035–1047.

[29] Daniel P. W. Ellis. "Beat Tracking by Dynamic Programming". In: *Journal of New Music Research* 36.1 (2007), pp. 51–60. ISSN: 0929-8215. DOI: 10 . 1080 / 09298210701653344.

[30] Stanley S. Stevens, John Volkmann, and Edwin B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190.

[31] Stanley S Stevens. "The Measurement of Loudness Level". In: *The Journal of the Acoustical Society of America* 27.5 (1955), pp. 815–829. ISSN: 0001-4966. DOI: 10.1121/1.1912650.

[32] Jonathan Foote. "Automatic audio segmentation using a measure of audio novelty". In: *International Conference on Multimedia and Expo* 1.C (2000), pp. 452–455. ISSN: 07320582. DOI: 10.1109/ICME.2000.869637.

[33] David Robinson. *ReplayGain specification.* 2014. URL: http://wiki.hydrogenaud.io/index.php?title=ReplayGain_specification (visited on 05/15/2017).

[34] Roman B Gebhardt and Jonas Margraf. "Applying Psychoacoustics to Key Detection and Root Note Extraction in EDM". In: *Proceedings of the 13th International Symposium on Computer Music Multidisciplinary Research (CMMR17), Matosinhos, Porto* (2017).

[35] Ángel Faraldo, Emilia Gómez, Sergi Jordà, and Perfecto Herrera. "Key estimation in electronic dance music". In: *Proceedings of the 38th European Conference on Information Retrieval Research (ECIR 2016), Padua, Italy* (2016), pp. 335–347.

[36] Dmitry Bogdanov et al. "Essentia: An Audio Analysis Library for Music Information Retrieval". In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013), Curitiba, Brazil* (2013), pp. 493–498.

[37] Beth Logan and Ariel Salomon. "A Music Similarity Function Based on Signal Analysis." In: *IEEE International Conference on Multimedia and Expo (ICME 2001)* (2001), pp. 22–25.

[38] Jean-Julien Aucouturier and Francois Pachet. "Music similarity measures: What's the use?" In: *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR 2002), Paris, France* (2002), pp. 13–17.

[39] Elias Pampalk, Simon Dixon, and Gerhard Widmer. "On the evaluation of perceptual similarity measures for music". In: *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx 2003), London, UK* (2003), pp. 7–12.

[40] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. "Music type classification by spectral contrast feature". In: *Proceedings of the IEEE International Conference on Multimedia and Expo* 1 (2002), 113–116 vol.1. DOI: 10.1109/ICME.2002.1035731.

[41] Vincent Akkermans, Joan Serrà, and Perfecto Herrera. "Shape-based spectral contrast descriptor". In: *Proceedings of the 6th Sound and Music Computing Conference (SMC), 23-25 July 2009, Porto, Portugal* (2009), pp. 143–148.

[42] Optical. *FABRICLIVE Promo Mix*. 2014. URL: https://soundcloud.com/fabric/optical-fabricliveprommix (visited on 05/15/2017).

[43] S.P.Y. *FABRICLIVE x Back To Basics Mix*. 2014. URL: https://soundcloud.com/fabric/spy-fabriclive-x-back-to-basics-mix (visited on 07/31/2018).

[44] Bass Brothers. *FABRICLIVE x Playaz Mix*. 2014. URL: `https://soundcloud.com/fabric/bassbrothers-fabriclive-x-playaz-mix` (visited on 07/31/2018).

[45] Adam L Berenzweig and Daniel PW Ellis. "Locating singing voice segments within music signals". In: *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the* (2001), pp. 119–122.

[46] Simon Leglaive, Romain Hennequin, and Roland Badeau. "Singing voice detection with deep recurrent neural networks". In: *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015), South Brisbane, Australia* (2015), pp. 121–125.

[47] Hanna Lukashevich, Matthias Gruhne, and Christian Dittmar. "Effective singing voice detection in popular music using arma filtering". In: *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07), Bordeaux, France* (2007).

[48] Mathieu Ramona, Gaël Richard, and Bertrand David. "Vocal detection in music with support vector machines". In: *Proceedings of the 33rd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008), Las Vegas, Nevada, USA.* (2008), pp. 1885–1888.

[49] Derry Fitzgerald. "Harmonic/percussive separation using median filtering". In: *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10), Graz, Austria* (2010).

[50] James L. Flanagan and Roger M. Golden. "Phase vocoder". In: *Bell Labs Technical Journal* 45.9 (1966), pp. 1493–1509.

[51] Werner Verhelst and Marc Roelands. "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech". In: *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1993), Minneapolis, MN, USA*

2.1 (1993), pp. 2–5. ISSN: 1520-6149. DOI: 10.1109/ICASSP.1993.319366.

[52] Jonathan Driedger, Meinard Muller, and Sebastian Ewert. "Improving time-scale modification of music signals using harmonic-percussive separation". In: *IEEE Signal Processing Letters* 21.1 (2014), pp. 105–109.

[53] Jonathan Driedger and Meinard Müller. "A Review of Time-Scale Modification of Music Signals". In: *Applied Sciences* 6.2 (2016), p. 57.

[54] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015. ISBN: 978-3-319-21944-8. DOI: 10.1007/978-3-319-21945-5.

[55] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. "madmom: a new Python Audio and Music Signal Processing Library". In: *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, Oct. 2016, pp. 1174–1178. DOI: 10.1145/2964284.2973795.

[56] Jouni Paulus, Meinard Müller, and Anssi Klapuri. "Audio-based music structure analysis". In: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010), Utrecht, the Netherlands* (2010), pp. 625–636.

[57] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), p. 27.

[58] Len Vande Veire. *Supplementary material: results of user test.* 2017. URL: https://docs.google.com/spreadsheets/d/e/2PACX-1vRpvbHqnvlwpG9qBlaHBth3_FNmu7HeD0Ycp6ePFQ7sN_COhBOQ9pfMt4eu5lwgHp1aQAPSMrI7lPUD/pubhtml (visited on 09/05/2017).

6

A CycleGAN for Style Transfer between Drum'n'bass Subgenres

The advent of advanced generative image-to-image translation networks [1, 2] has led to interesting musical applications, e.g. the transfer of timbre between individual instruments [3]. In the work presented in this Chapter, I apply the CycleGAN image-to-image translation framework to Mel-scaled log-amplitude spectrograms. This leads to a successful transfer of audio texture between audio segments of two subgenres of drum'n'bass music, namely *liquid* and *dancefloor* drum'n'bass. This effectively creates a rudimentary *remix* of the original segment in the other subgenre. It hence offers a new creative tool for DJs and music producers, e.g. to quickly prototype a remix of an existing song in another genre, or to use as an effect in live performances.

My contribution is two-fold. Firstly, I show that a meaningful transfer of musical texture can be realized with relatively little

training data and computational resources. Secondly, when reconstructing the spectral phase for the generated amplitude spectrogram, I avoid audio degradation – as observed when using existing algorithms for phase reconstruction – by simply using the phase of the original audio recording. This shows that this is a simple yet effective heuristic for this purpose.

> *The research presented in this chapter has been presented at the Machine Learning for Music Discovery Workshop (ML4MD) at the 36th International Conference on Machine Learning (ICML 2019). The corresponding publication is:*
>
> *Vande Veire L., De Bie T., Dambre J. A CycleGAN for style transfer between drum and bass subgenres. In Proceedings of the Machine Learning for Music Discovery Workshop (ML4MD) at the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA, 2019 [4] .*

6.1 Background: unpaired image-to-image translation using cycle-consistent adversarial networks

The goal of image-to-image translation is to transform images from one "image domain" into another. Examples include the translation of natural images to paintings in the style of a particular artist, transforming images of winter landscapes into images of summer landscapes, or turning images of horses into images of zebras. Cycle-Consistent Adversarial Networks [1] – or CycleGAN in short – are a technique that allow to learn such transformations. Traditional image-to-image translation methods often require a dataset of aligned image pairs in order to learn the desired mapping. CycleGAN improves upon this by allowing to learn these mappings in the absence of paired examples. Instead, one only needs a collection of (unpaired) images in both image domains.

More formally, CycleGAN learns a mapping $G : X \rightarrow Y$ from image domain X to image domain Y, where G is defined as a

deep neural network. At the same time, the inverse transformation $F : Y \rightarrow X$ is learned. In order to ensure that the transformed images (respectively $G(x)$ and $F(y)$ with $x \in X$ and $y \in Y$) look like realistic images from the respective output domains, two discriminator networks D_X and D_Y are trained alongside G and F in an adversarial framework. In this framework, an *adversarial loss function* expresses that the discriminators should detect as well as possible whether images are "real" or "fake", i.e. D_Y attempts to detect whether an image y was drawn directly from domain Y or whether it was generated by G (D_X does the same for image domain X and generator network F). The network parameters of the discriminators are then updated in order to minimize this loss function. However, at the same time, the parameters of the generator networks G and F are updated in order to maximize it. In other words, the generators are trained to "fool" the discriminators by generating images that look as realistic as possible. This procedure effectively allows the system to learn to generate realistic images.

However, the generators should create not just any realistic-looking image in the output domain, but rather an image that is the "translation" of the input image. To do so, a *cycle consistency loss* is imposed. In essence, this loss function expresses that the inverse transformation of a translated image should be identical to the original input image, i.e. $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Optimizing the generators and discriminators in conjunction to simultaneously optimize the adversarial loss and cycle consistency loss functions allows to learn an image-to-image translation system using only unpaired data. For more details about the CycleGAN framework, I refer to the original publication by Zhu & Park et al. [1].

6.2 Methods

6.2.1 Dataset description

The spectrograms for training the CycleGAN model are extracted from 40 *liquid* and 40 *dancefloor* drum'n'bass songs. From each

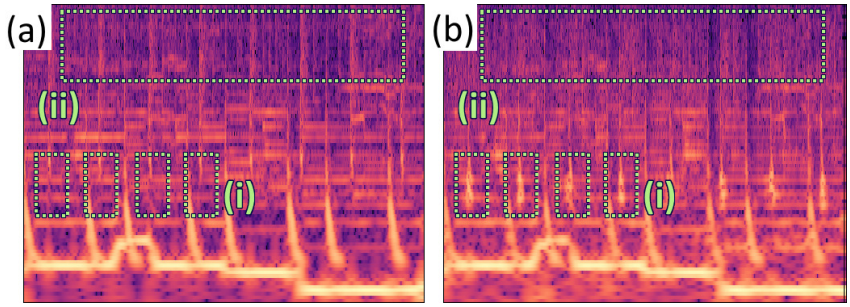


Figure 6.1: Side-by-side comparison of an input extract and the corresponding output. This figure shows CQT spectrograms instead of Mel-scaled spectrograms as the higher resolution of the CQT transform for lower frequencies illustrates the observed CycleGAN transformations more clearly. (a) input audio segment (genre: *'liquid'*). (b) output audio segment (genre: *'dancefloor'*). Annotated style transfer elements: (i) examples of the introduction of a snare drum; (ii) insertion of noise after hi-hats in high frequency regions.

song, 3 segments of 4 downbeats (each approximately 5.5 seconds long) are selected, resulting in a total of 240 segments for training. The segments are downbeat-aligned and selected from the 'main' part of the song (similar to the chorus for vocal-based music). The tempo of each song is stretched to 175 BPM using WSOLA time stretching [5] to ensure a consistent length for training. Note that after training, the model can be applied to extracts in any tempo.

6.2.2 CycleGAN model training

The CycleGAN model implementation by Zhu and Park [6] is used for this work. The model (with 9 residual blocks) operates on Mel-scaled spectrogram images. This representation is obtained by first transforming each audio segment into a time-frequency spectrogram S using a STFT with 2048 frequency bins and a hop size of 512. Then, the squared amplitude of that spectrogram is calculated and converted to a decibel scale ($10 \log_{10}(|S|^2)$). This is then trans-

formed to a Mel-frequency scale [7], to offer a higher resolution in the lower frequency ranges. Finally, the spectrogram values are normalized between 0 and 1, and saved as grayscale images to be processed by the CycleGAN model. The network finishes training within a few hours on a single GeForce GTX 1080 GPU.

6.2.3 Audio reconstruction from amplitude spectrogram

Reconstructing the audio from a generated spectrogram first involves reverting the preprocessing steps: the initial pixel values are scaled back to the empirically observed average log-amplitude range of -15 dB to 65 dB, and the inverse Mel-frequency and log-amplitude transformations are applied to obtain the STFT amplitude spectrogram. The phase of the spectrogram is approximated by the phase of the original audio segment. Applying the inverse STFT yields the audio reconstruction.

6.3 Results and discussion

Figure 6.1 shows the side-by-side comparison of two spectrograms: an original spectrogram, and the result after applying the trained CycleGAN. The corresponding audio fragments and additional examples are available online[1].

When translating a *liquid* segment to *dancefloor* drum'n'bass, several observations stand out. Firstly, the network inserts snare drums on every second beat of each measure, or changes the sound of already existing snare drums on those beats, giving them a harder and more 'punchy' sound ((i) in Figure 6.1). Secondly, noise is added in the high frequency regions, changing the timbre of the hi-hats ((ii) in Figure 6.1). We also observe a slight amplitude reduction of melody components in the mid/high frequency ranges.

In the reverse transformation (*dancefloor* to *liquid*), the reverse operations are observed – snare drums are less pronounced and they

[1] `https://lenvdv.github.io/papers/2019-cyclegan-examples/`

sound 'brighter', hi-hats are accentuated by reducing noise around them, and melody components are emphasized differently. We refer to the online examples for spectrograms and audio excerpts.

A major challenge in reconstructing high-quality audio is predicting the phase for the amplitude spectrograms. Our implementation simply uses the phase of the input spectrogram as an approximation. Existing phase reconstruction techniques [8, 9] were found to introduce a significant low-frequency 'buzzing' sound in the audio. This appears to be an undesired consequence of using the Mel-scale transformation, which smears out the spectral energy between adjacent frequency bands. After applying the phase reconstruction algorithms, this leads to interfering frequency components and hence a degraded audio signal. Directly modeling the spectrogram phase (or rather, the instantaneous frequency) in the CycleGAN as in Engel et al. [10] did not yet lead to the intended results, and is currently left as a path for future exploration.

6.4 Conclusions

We presented a method to translate segments between two subgenres of drum'n'bass, by applying the CycleGAN framework to Mel-scaled log-amplitude spectrograms. To reconstruct the audio, we approximate the phase of the generated amplitude spectrogram by the phase of the input spectrogram, avoiding artifacts that are introduced by existing phase reconstruction methods. This work could be a first step towards automated remix generation, and exploring more genre combinations and alternative network architectures better tuned to the properties of music spectrograms offer interesting opportunities for further research.

References

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV 2017), Venice, Italy*. 2017.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA*. 2017.

[3] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B. Grosse. "TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA*. 2019. URL: `https://openreview.net/forum?id=S1lvm305YQ`.

[4] Len Vande Veire, Tijl De Bie, and Joni Dambre. "A CycleGAN for style transfer between drum and bass subgenres". In: *Proceedings of the Machine Learning for Music Discovery Workshop (ML4MD) at the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA*. 2019.

[5] Werner Verhelst and Marc Roelands. "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech". In: *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. IEEE. 1993, pp. 554–557.

[6] Jun-Yan Zhu, Taesung Park, and Tongzhou Wang. *CycleGAN and pix2pix in PyTorch*. 2017. URL: `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

[7] Stanley S. Stevens, John Volkmann, and Edwin B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893. eprint: https://doi.org/10.1121/1.1915893. URL: https://doi.org/10.1121/1.1915893.

[8] Daniel Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243.

[9] Zinglei Zhu, Gerald T. Beauregard, and Lonce Wyse. "Real-Time Iterative Spectrum Inversion with Look-Ahead". In: *2006 IEEE International Conference on Multimedia and Expo (ICME 2006), Toronto, ON, Canada*. 2006, pp. 229–232. DOI: 10.1109/ICME.2006.262424.

[10] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. "GANSynth: Adversarial Neural Audio Synthesis". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA* (2019). URL: https://openreview.net/forum?id=H1xQVn09FX.

Part III

Conclusions and Perspectives

7 Conclusions and Perspectives

In this chapter, I provide a summary of the most important research conclusions from this thesis. I also discuss potential directions for further research. More specifically, I will explain my view on future challenges in the context of automatic drum transcription and in the context of developing music processing systems using MIR technology.

7.1 Conclusions

In Part I, I investigated the usage of non-negative matrix factor deconvolution for automated drum mixture decomposition. NMFD is an interesting algorithm for this task because it does not require a large dataset to train on, making it easy to apply to drum recordings with uncommon instruments. NMFD also performs a transcription while at the same time extracting a template of each distinct drum sound. This property enables many interesting applications, ranging from musicological analysis [1] and the

automatic resampling of drum sounds [2], to browsing through or querying a music database based on drum sample properties.

In this thesis, I focused on addressing two issues with NMFD. Firstly, I illustrated that the activations that are discovered by NMFD are not impulses; instead, they are often spread out over time. This does not make sense in the context of a drum mixture, where instantaneous activations would be expected. It also contradicts the interpretation that NMFD describes a mixture as the repetition of short spectrogram templates through time. Instead, individual sounds in the mixture are often decomposed as the sum of several overlapping templates, making it hard to discern where in the mixture a certain drum hit occurs. The solution that I proposed in Chapter 3 effectively leads to sparse and impulse-like activation patterns, which results in more interpretable decompositions.

The second issue that I addressed is the tendency of NMFD to capture sequences of drum strokes in one template, instead of a single drum hit, when the length of the templates is relatively long. Note that often this issue cannot be resolved by simply choosing a shorter template length: some recordings contain drum hits with a relatively long decay and consequently require long templates to be appropriately modeled. Instead, in Chapter 4, I proposed an ad-hoc modification that detects the emergence of secondary onsets in the templates during optimization, and that overwrites any excess drum hits in a template as soon as they start to appear. I showed that this indeed reduces the number of "double hits" in the discovered templates.

In Part II, I focused on using MIR techniques to interact with music in innovative ways. I focused on the genre of drum'n'bass in particular. In Chapter 5, I described the implementation of an automated DJ system for drum'n'bass music. This work illustrates that, by combining several MIR techniques, it is possible to obtain

an automated DJ system with a high annotation accuracy of the target music genre. The system is able to achieve such a high performance partially thanks to the fact that the methods are tailored to the target genre, which advocates for genre-specific MIR systems if a high accuracy of annotation is desired. Through a user test, I illustrated that guiding the mixing process using timbrical features is important to obtain an enjoyable DJ mix.

In Chapter 6, I demonstrated the realization of a music style transfer system for drum'n'bass. This system uses an unpaired image-to-image translation technique based on a Generative Adversarial (neural) Network (GAN) model applied to music spectrograms. The system more specifically learns (without supervision) to adapt the timbre of the drum hits to the target genre. Learning this transformation was possible with relatively little training data (120 training segments for each domain or 240 segments in total). This offers the exciting perspective that interesting musical applications, such as specific genre transformation effects, can be achieved quite easily and without large amounts of training data.

7.2 Future Perspectives

In what follows, I discuss what I consider to be interesting avenues for further research. Section 7.2.1 discusses my view on further research in the context of ADT using the NMFD algorithm. Section 7.2.2 continues with a discussion of future work in the field of ADT in general. Finally, in Section 7.2.3, I discuss my perspective on the further development of systems that realize novel ways to experience and make music.

7.2.1 Perspectives for Drum Mixture Decomposition using the NMFD Paradigm

In essence, I find NMFD interesting for transcription applications mostly because of the underlying paradigm, i.e. the alternation

between improving estimates of the source sounds and improving the estimated locations of those sounds. The algorithm thus simply exploits the repetitive nature of the sounds in the recording, without imposing any constraints on what those sounds could be. This makes the algorithm inherently capable of transcribing mixtures with uncommon instruments.

Unfortunately, NMFD is imperfect and often converges to undesired solutions, for example with non-impulse-like activations or with templates capturing sequences of drum strokes instead of individual drum hits. NMFD is too flexible on the one hand – e.g. there are no constraints to what can be captured within a template or what the activations should look like – and on the other hand it is not flexible enough – e.g. templates are supposed to repeat unaltered through time, and the entire mixture must be modeled without leaving any residuals. I therefore propose three ways to find a better balance between flexibility and rigidity in the NMFD framework. Firstly, NMFD (or an algorithm following the NMFD paradigm) needs to be more discriminative with respect to the kind of sounds that it captures in the templates. Secondly, NMFD should allow some variation in how these templates are instantiated through time in the mixture. Finally, there should be a mechanism to perform only partial decompositions. In what follows, these proposals are explained in more detail.

Firstly, the algorithm needs to have built-in mechanisms to bias the templates to capture musically meaningful single-hit percussive sounds. NMFD currently has no regard for the musical validity of the discovered templates, as illustrated for example by the double-hit problem discussed in Chapter 4. Other issues that can occur are for example that the templates capture only part of an instrument (e.g. only the attack, only the low frequencies, ...). In this case, two or more templates are needed to model one drum hit, leading to a fragmented decomposition. A solution would be to incorporate some knowledge in the algorithm about what

a single drum hit should sound like. This could be realized for example by incorporating a machine learning model that detects whether a short spectrogram extract represents a "realistic" drum hit or not, and consequently steers the algorithm towards more meaningful templates. Note that this makes the algorithm reliant on a (large-scale) dataset of individual drum sounds, limiting the advantages of NMFD over deep learning-based ADT methods. Another approach would be to incorporate human feedback in the decomposition loop. The user could provide a coarse annotation of the approximate shape and location of some of the individual drum sounds in the spectrogram in order to guide the subsequent decomposition process.

Secondly, I propose to relax the assumption that drum sounds appear in a spectrogram as exact copies of the templates (scaled by some amplitude factor). This assumption makes the algorithm unable to appropriately deal with slight differences in e.g. timbre or decay time due to a slightly different playing technique, a varying velocity, or music production techniques used in the mixing process such as compression, side-chain compression or dynamic equalization. Note that the aforementioned assumption might not hold even if the sounds in a recording (i.e. represented as a waveform) are identical: if two identical drum hits align differently with the analysis frames of the time-frequency transformation, then they will look ever so slightly different in the resulting spectrogram. One could argue that these distortions are sufficiently small to ignore without compromising the decomposition quality. If certain sounds are still significantly different due to one of the aforementioned effects, then another template could be introduced to remedy any resulting issues. This is the approach that I followed in this thesis and it seems to work reasonably well in most cases. However, on a conceptual level, these workarounds are flawed, and they might be too limiting in applications where the aforementioned effects do matter and where one wishes to capture these nuances. Further research could therefore look into how the

(instantiations of the) templates could become more flexible. For example, the templates could be redefined as a (parameterized) generative model of drum spectrograms. Each instantiation of the template is then described not only by a peak in the activation function, but also by a set of parameters for the generative model that capture small variations in the drum hit. Note that redefining the templates using a parameterized model of drum hits is analogous to the proposal from the previous paragraph to capture more realistic drum hits in the templates, so these ideas go hand in hand.

Thirdly, I hypothesize that NMFD is limited by its objective to decompose the complete spectrogram. If a spectrogram contains elements that cannot be modeled appropriately (e.g. non-percussive sounds, or variations in the sounds caused by any of the aforementioned effects), then the objective function still incentivizes the algorithm to try to model these elements anyways. This leads to artifacts in the decomposition. It would instead be much more interesting to have a mechanism to detect which parts of the spectrogram approximately match with the templates and to ignore parts of the spectrogram that cannot (yet) be modeled meaningfully. The output is then a partial decomposition of the spectrogram, with transcriptions and refined templates for the "matching" parts of the spectrogram and a residual signal containing all that could not be modeled (note that this resembles a combination of music sound source separation and automatic drum mixture decomposition).

7.2.2 Perspectives for Automatic Drum Transcription

This section outlines my perspective on future directions for ADT research beyond NMFD.

One commonly raised issue in the ADT field is that publicly available datasets are often limited both in size as well as in instrument variety [3, 4]. Consequently, many ADT methods focus

on transcribing only the three predominant acoustic percussive instruments in Western music, i.e. the kick drum, snare drum and hi-hat [4]. Progress is being made in that regard, though, by releasing new datasets [5, 6] and by training models to transcribe more instruments [4].

While this progress is encouraging, one concern that I have is that most contemporary ADT systems are rigid by design, because they define the considered instrument classes at the time of implementation. Furthermore, which sounds will be detected as belonging to each class is defined by the particular dataset used to train the (machine learning) ADT system. I believe that this introduces a risk of "pigeonholing" percussive sounds into categories that are stipulated by the developers of the ADT systems or by the datasets used to develop these systems, but that are less relevant to realistic transcription applications. This consequently limits the practical applicability of these ADT systems. For example, while this approach could be well-suited to transcribe recordings containing a "standard" traditional acoustic drum kit, it might be less appropriate for e.g. transcribing synthesized drum sounds in electronic music, where the percussive sounds might not fit into these predefined and fixed categories. Therefore, I think that further research on more flexible and instrument-class-agnostic transcription methods based on e.g. NMF(D) or few-shot learning [7] could be very interesting. These methods could then adapt with little additional data to the specific requirements of novel use cases. A few-shot model still needs a human to provide a few examples of each instrument class, though. To get the human out of the loop, it would be interesting to look into incorporating this few-shot transcription mechanism into the NMFD paradigm (see Section 7.2.1).

Finally, in the development and evaluation of ADT systems, a distinction is often made between transcribing recordings containing only the target instruments, transcribing mixtures containing

other percussive instruments besides the instruments that are to be transcribed, and transcribing recordings that also contain non-percussive instruments [3]. However, advances in music source separation with U-Net-based technologies and the open-source release of readily available high quality separation tools such as Spleeter [8] and OpenUnmix [9] may lead to a shift in the ADT paradigm: if one can reliably extract a high-quality version of the percussive-only signal, does it then still make sense to develop ADT systems for multi-instrumental systems? Instead, one could perform a source separation step first, and then transcribe the outcome. This allows to have more modular systems, where one can easily swap out one separation system for a better one when that becomes available. It also allows the ADT community to focus on the development of ADT with percussive-only signals, potentially leading to a faster development of high-quality transcription systems that can deal with multiple percussive instruments, systems that adapt to the requirements of the user or use case etc. There still is a case to be made for transcription systems that can deal with multi-instrumental mixtures, though. For example, in the proposed modular design, the ADT system could be limited by the performance of the source separation system. There also is the argument of processing speed: a system that performs the transcription directly could be much faster than a system that has to separate the percussive signal from the mixture first. This is important for time-sensitive applications such as real-time transcription in a live setting or for interactive applications. Nevertheless, the improvements of automatic music transcription systems and of music source separation systems go hand in hand: better separation systems can provide a solid foundation for transcription and better transcription systems may provide a good signal for separation systems in order to separate the target mixture into its constituent components.

7.2.3 Perspectives for Automatic Music Generation and Modification Systems

In Chapter 5 and Chapter 6, I presented two MIR systems that automatically perform non-trivial and creative musical audio transformations, more specifically an automated DJ system and a style transfer system for drum'n'bass.

Looking back on these projects, it is important to acknowledge their artistic and therefore subjective nature. Throughout their development, I namely made several design decisions that have shaped their creative outcome. For example, in the automated DJ system, I decided to limit the scope of the project to drum'n'bass music; I chose to implement a particular mixing style for this genre of music – namely a mix with *seamless* transitions and a smooth progression between songs – and I consequently designed a song selection process and transition types that work towards this goal. For the style transfer system, the most important artistic decision is the choice of the two subgenres to transform between, and especially what songs I selected as training examples for the varieties of "liquid" and "dancefloor" drum'n'bass that I had in mind. The artistic nature of these decisions of course implies that many other choices are possible, which could lead to radically different creative systems. For example, instead of seamless mixes, one could aim to implement an automated DJ system that creates mixes full of musical contrast, tension and friction. This again leads to many new creative choices to make: for example, how does one understand musical tension and friction in the context of a DJ mix? How could it be established, which aspects of the music are used to create this contrast? How are these processes and concepts abstracted into algorithms and features? In the case of the style transfer system, a different choice of training examples for each drum'n'bass subgenre could result in a completely different transformation learned by the system (the system would perhaps have learned to modify the timbre of the bass instead of that

of the drums). This is possible because the subgenres of liquid and dancefloor drum'n'bass are not rigorously defined and also encompass many overlapping flavors of music ("sub-subgenres", if you will). This results in some artistic freedom when choosing a set of representative training examples for each subgenre. Because of their artistic character, the presented works should not be understood as a step towards some complete and universal systems to perform certain musical transformations automatically (DJing, creating remixes). Instead, they are demonstrations of the capabilities of MIR technology to realize non-trivial music transformations, in one of the many and diverse ways in which those transformations can be performed.

This perspective of the MIR-system-as-an-artistic-project also highlights the role that the interpretation and abstraction of musical concepts play in the development of MIR systems. This is reminiscent of the concept of the "semantic gap" that was discussed in Section 1.1.2. For example, the automated DJ system from Chapter 5 implements a three-dimensional numerical "style descriptor" feature in order to capture the general style and atmosphere of each track. While this leads to a measurable quantity that is convenient to work with, this process also reduces the multi-faceted and subjective concept of music similarity to a numerical feature that is very limited in scope and biased because of the way it is constructed. Similarly, the CycleGAN style transfer system uses an adversarial cost function in its optimization, which effectively means that it uses a (machine-learned) metric that "measures" to what extent a spectrogram of a piece of music belongs to the target subgenre. This approach tacitly assumes that one can quantitatively measure how much a music excerpt belongs to a certain genre; as such, it reduces the complex and fuzzy concept of genre to a convenient but simplistic quantitative feature. This does not mean that these models are not useful or meaningful. For example, from informal tests and the results of the subjective user study, it is clear that the style descriptor feature

of the automatic DJ system works quite well for the intended song selection algorithm. Also the CycleGAN model is effective in learning musically interesting and meaningful transformations. However, this observation does mean that one should not lose sight of the fact that these features and models are nothing but a proxy for the actual musical concepts they attempt to describe, and that they will most likely not capture those concepts in all their detail, depth and nuance. The same goes for the evaluation of MIR systems by means of "objective" numerical metrics: one should keep in mind that some aspects of the performance of an algorithm can be hard to measure or compare with other algorithms, simply because the underlying musical concepts are hard to objectively quantify. In short, this shows that the semantic gap is pervasive in the world of MIR, and one should keep this in mind when designing and evaluating MIR systems.

As described in the introduction of this thesis, music making tools based on advanced MIR techniques are becoming more widespread, e.g. as commercial products tailored to both unexperienced music enthusiasts and expert music producers¸ or as a part of new projects by established artists. This adoption of MIR technology is what I find extremely exciting, and I believe that in the coming years these technologies could transform how we interact with music.

Firstly, I believe that music making will become more accessible thanks to music generation and manipulation technologies. Generative systems can support artists in quickly realizing their ideas, for example by offering suggestions for melodies or rhythms, or by generating musical elements to quickly flesh out an existing musical idea. This also empowers people without a strong musical background to create, so that the art of making music becomes accessible to everyone, and not only to the lucky few with enough time and money to dedicate to a music education and to building up experience. There are a few caveats in the realization of such

systems, though, as discussed in Holzapfel et al. [10]. For example, at the moment, generative MIR technologies (and MIR technologies in general) are often biased towards Western popular music, typically because this music is most easily available and of most interest to the developers or to the communities in which those developers live and disseminate their work. This could unduly nudge music makers towards adapting their music to fit this Western paradigm, simply because the novel MIR technology that they want to use doesn't work as well with non-Western music. I agree with their conclusion that the biases in and the consequential limitations of MIR work should become more explicit. Additionally, it would help to have more diversity in datasets and MIR technologies that can be adapted easily without larger engineering expertise. Another caveat mentioned by Holzapfel et al. is that these generative technologies could trivialize and commoditize the art of music making. Therefore, I think that MIR technology developers should reflect on how (and subsequently act so that) their technology is in service of the music and the artist, rather than a technology that creates an easily accessible substitute for music making at the expense of the richness and variety in the musical landscape[1].

I furthermore think that the most exciting and creative applications will not emerge purely thanks to technological or academic experimentation and developments, but only after artistic experimentation by many artists and creative communities. It is therefore extremely important that these new technologies are accessible to the musicians who would explore these new artistic avenues. In order to realize that, it is essential that the scientific community interacts and collaborates with artists "in the field". In the con-

[1] I don't think that we would descend into a dystopia of bland, auto-generated, easily-digestible pop music if MIR technology would gain traction in this way, though: I am convinced that musicians will be inventive and rebellious enough to counter any such evolution by responding with their own provocative and musically rich creations. Instead, my argument is that I would much rather want to see MIR technology be a part of the musical richness, rather than an adversary.

text of this thesis, it would for example be interesting to study how music producers and DJs use ADT technology, what they need in their creative process, and how these needs can be fulfilled with ADT tools. This could illuminate what aspects of current ADT systems limit their practical usability, which is important to identify directions for further research. Similarly, it would be interesting to see research on the best practices of DJs and how MIR technology could fit into their artistic process. These are open questions not addressed in this thesis, and I believe that they are valuable directions for further research.

However, I believe that as the MIR community, we should go one step further than this: it should be possible for musicians to get started with MIR technologies, to tweak and experiment with them, without having to interact with the scientific community that developed them and without needing in-depth technical knowhow or sophisticated infrastructure. In other words, accessibility of these technologies is key. There already seems to be an awareness of this within the MIR community. For example, there have been many initiatives to collaborate with artists and to release MIR technology in formats that are familiar to music producers, e.g. as plugins for digital audio workstations such as Ableton Live or FL Studio [11]. The emergence of several web-technology-based (JavaScript) MIR- and machine learning frameworks such as Essentia.js [12] and Magenta.js [13] also fits within this picture, as these allow the development of interesting prototypes with relative ease that are furthermore accessible through a modern web browser.

In conclusion, music analysis, generation and manipulation technologies are becoming increasingly more powerful, and they are receiving attention and adoption by both artistic communities as well as by commercial players. I believe that we will see many exciting and perhaps unexpected uses of these technologies that could cause major changes in how we create and experience music. Whether they will blend smoothly into the current paradigm – allowing us to do what we already can faster, easier or otherwise

"better" – or whether they will cause a full paradigm shift and the emergence of completely new ways in how we make and experience music, remains an open question. Regardless, I am incredibly excited to have contributed to this field, and I am looking with curiosity to what the future will bring.

References

[1] Christian Dittmar and Meinard Müller. "Reverse Engineering the Amen Break – Score-Informed Separation and Restoration Applied to Drum Recordings". In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 24.9 (2016), pp. 1531–1543. ISSN: 23299290. DOI: 10.1109/TASLP.2016.2567645.

[2] Patricio López-Serrano, Matthew E. P. Davies, Jason Hockman, Christian Dittmar, and Meinard Müller. "Break-Informed Audio Decomposition For Interactive Redrumming". In: *19th International Society for Music Information Retrieval Conference (ISMIR 2018) - Late-Breaking/Demos Session, Paris, France* (2018).

[3] Chih-wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, and Meinard Müller. "A Review of Automatic Drum Transcription". In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 26.9 (2018), pp. 1457–1483. DOI: 10.1109/TASLP.2018.2830113.

[4] Richard Vogl, Gerhard Widmer, and Peter Knees. "Towards Multi-Instrument Drum Transcription". In: *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx-18), Aveiro, Portugal* (2018).

[5] Lee Callender, Curtis Hawthorne, and Jesse Engel. "Improving Perceptual Quality of Drum Transcription with the Expanded Groove MIDI Dataset". In: *arXiv preprint arXiv:2004.00188* (2020).

[6] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. "Drum Transcription via Joint Beat and Drum Modeling using Convolutional Recurrent Neural Networks". In: *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017), Suzhou, China* (2017), pp. 150–157.

[7] Yu Wang, Justin Salamon, Mark Cartwright, Nicholas J Bryan, and Juan Pablo Bello. "Few-Shot Drum Transcription in Polyphonic Music". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR), Montréal, Canada* (2020).

[8] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020). Deezer Research, p. 2154. DOI: 10.21105/joss.02154. URL: https://doi.org/10.21105/joss.02154.

[9] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. "Open-Unmix - A Reference Implementation for Music Source Separation". In: *Journal of Open Source Software* (2019). DOI: 10.21105/joss.01667. URL: https://doi.org/10.21105/joss.01667.

[10] Andre Holzapfel, Bob L. Sturm, and Mark Coeckelbergh. "Ethical Dimensions of Music Information Retrieval Technology". In: *Transactions of the International Society for Music Information Retrieval* 1.1 (2018), pp. 44–55.

[11] Google Magenta. *Magenta Studio (v1.0)*. 2021. URL: https://magenta.tensorflow.org/studio (visited on 03/25/2021).

[12] Albin Andrew Correya, Dmitry Bogdanov, Luis Joglar-Ongay, and Xavier Serra. "Essentia.js: a JavaScript library for Music and Audio Analysis on the Web". In: *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020), Montréal, Canada*. 2020.

[13] Adam Roberts, Curtis Hawthorne, and Ian Simon. "Magenta.js: A JavaScript API for Augmenting Creativity with Deep Learning". In: *Joint Workshop on Machine Learning for Music (ML4MD) at ICML 2018, Stockholm, Sweden*. 2018.

Appendices

A

Derivation of the Update Rules for Sigmoidal NMFD

The sigmoidal NMFD model with saturating activations is defined by Equation (3.2). Note that the logit-activations $G_{k,t}$ and the logit-amplitudes a_k can take negative values. The objective function for the optimization, $\mathcal{L}_{\text{tot}}$, is given by Equation (3.6), with its constituent terms $\mathcal{L}_{\text{KL}}$ and $\mathcal{L}_{\text{G}}$ given Equations (3.3) and (3.4), respectively. The inverse of the logistic function, σ^{-1}, is given by

$$\sigma^{-1}(y) = \ln\left(\frac{y}{1-y}\right). \tag{A.1}$$

A.1 Additive Gradient-Descent Update for G

G minimizes $\mathcal{L}_{\text{tot}}$ using gradient-descent, see Equation (3.8). The derivative of the logistic function, Equation (3.1), with respect to

its argument, is

$$\frac{d\sigma(x)}{dx} = \frac{\exp(-x)}{(1+\exp(-x))^2} = \sigma(x)(1-\sigma(x)) = \sigma(x)\sigma(-x). \quad \text{(A.2)}$$

From Equations (3.3) and (A.2), the partial derivative $\frac{\partial \mathcal{L}_{\mathrm{KL}}}{\partial G_{k,t}}$ is given by

$$\frac{\partial \mathcal{L}_{\mathrm{KL}}}{\partial G_{k,t}} = \sum_{n,\tau} \left(1 - \frac{X_{n,t+\tau}}{\hat{X}_{n,t+\tau}}\right) \sigma(G_{k,t})\sigma(-G_{k,t}) W^{(k)}_{n,\tau} \sigma(a_k). \quad \text{(A.3)}$$

From Equation (3.4), $\frac{\partial \mathcal{L}_G}{\partial G_{k,t}}$ can be calculated:

$$\begin{aligned}
\frac{\partial \mathcal{L}_G}{\partial G_{k,t}} &= \frac{\partial}{\partial G_{k,t}} \sum_{k',t'} \exp\left(-\left(\frac{G_{k',t'} - \mu_{k'}}{2}\right)^2\right) && \text{(A.4)}\\
&= \frac{\partial}{\partial G_{k,t}} \left[\exp\left(-\left(\frac{G_{k,t} - \mu_k}{2}\right)^2\right)\right] \\
&\quad + \frac{\partial}{\partial G_{k,t}} \left[\sum_{t' \neq t} \exp\left(-\left(\frac{G_{k,t'} - \mu_k}{2}\right)^2\right)\right] && \text{(A.5)}\\
&= -\left(G_{k,t} - \mu_k\right) \exp\left(-\left(\frac{G_{k,t} - \mu_k}{2}\right)^2\right) \\
&\quad + \sum_{t'} \left(G_{k,t'} - \mu_k\right) \exp\left(-\left(\frac{G_{k,t'} - \mu_k}{2}\right)^2\right) \frac{\partial \mu_k}{\partial G_{k,t}}. && \text{(A.6)}
\end{aligned}$$

The right-hand term in Equation (A.6) is non-zero when $G_{k,t}$ is used in the calculation of μ_k, see Equation (3.5):

$$\frac{\partial \mu_k}{\partial G_{k,t}} = \begin{cases} \frac{\alpha_k}{\sigma(\mu_k)\sigma(-\mu_k)} \sigma(G_{k,t})\sigma(-G_{k,t}), & G_{k,t} = \max_{t'}(G_{k,t'}) \\ \frac{1-\alpha_k}{\sigma(\mu_k)\sigma(-\mu_k)} \sigma(G_{k,t})\sigma(-G_{k,t}), & G_{k,t} = \min_{t'}(G_{k,t'}) \\ 0, & \text{otherwise} \end{cases}, \quad \text{(A.7)}$$

in which we used the derivative of the logit function, Equation (A.1):

$$\begin{aligned}\frac{d\sigma^{-1}(y)}{dy} &= \frac{1}{y(1-y)} \\ &= \frac{1}{\sigma(x)(1-\sigma(x))} \\ &= \frac{1}{\sigma(x)\sigma(-x)}, \text{ with } y = \sigma(x). \end{aligned} \tag{A.8}$$

Note that the left-hand term in Equation (A.6) is always negative when $G_{k,t} > \mu_k$ and always positive when $G_{k,t} < \mu_k$. In the gradient-descent updates, Equation (3.8), this will always cause $G_{k,t}$ to grow away from μ_k, i.e., it pushes $G_{k,t}$ towards saturation.

However, when $G_{k,t} = \min_{t'}(G_{k,t'})$ or $G_{k,t} = \max_{t'}(G_{k,t'})$, the right-hand term in Equation (A.6) can have the opposite sign of the left-hand term, potentially canceling it out or even updating $G_{k,t}$ in the other direction, i.e., *away* from saturation. $\mathcal{L}_G$ is then minimized not by pushing the activations towards saturation, i.e., moving $G_{k,t}$ away from μ_k, but instead by moving μ_k away from $G_{k,t'}$. This might lead to unstable updates where the ultimate values of $G_{k,t}$ are hindered from growing to saturation, an undesired effect which we wish to prevent. Therefore, we regard μ_k as a constant when applying the updates, i.e., we ignore the right-hand term in Equation (A.6). This gives the expression from Equation (3.10) for the derivative of $\mathcal{L}_\mathrm{G}$ with respect to $G_{k,t}$:

$$\frac{\partial \mathcal{L}_G}{\partial G_{k,t}} \approx -\left(G_{k,t} - \mu_k\right) \exp\left(-\left(\frac{G_{k,t} - \mu_k}{2}\right)^2\right). \tag{A.9}$$

The expression is exact for all $G_{k,t}$, except when $G_{k,t} = \max_{t'}(G_{k,t'})$ or $G_{k,t} = \min_{t'}(G_{k,t'})$.

A.2 Multiplicative Update for W

The derivation of the multiplicative updates for $W^{(k)}_{n,\tau}$ is analogous to the derivation in Schmidt and Mørup [1] and Lee and Seung [2]. Consider the gradient-descent updates for $W^{(k)}_{n,\tau}$ that minimize $\mathcal{L}_{\text{tot}}$ with learning rate η:

$$W^{(k)}_{n,\tau} \leftarrow W^{(k)}_{n,\tau} - \eta \frac{\partial \mathcal{L}_{\text{tot}}(X, \hat{X}, G)}{\partial W^{(k)}_{n,\tau}}. \tag{A.10}$$

If we rewrite $\frac{\partial \mathcal{L}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}$ as the difference of two strictly positive terms $\frac{\partial \mathcal{L}^{+}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}$ and $\frac{\partial \mathcal{L}^{-}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}$,

$$\frac{\partial \mathcal{L}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}} = \frac{\partial \mathcal{L}^{+}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}} - \frac{\partial \mathcal{L}^{-}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}, \tag{A.11}$$

then we can choose $\eta = W^{(k)}_{n,\tau} / \frac{\partial \mathcal{L}^{+}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}$ as in Lee and Seung [2] and Schmidt and Mørup [1] so that the first term in Equation (A.10) cancels out. This gives

$$W^{(k)}_{n,\tau} \leftarrow W^{(k)}_{n,\tau} \frac{\frac{\partial \mathcal{L}^{-}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}}{\frac{\partial \mathcal{L}^{+}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}}}. \tag{A.12}$$

The derivative of $\mathcal{L}_{\text{tot}}(X, \hat{X}, G)$ with respect to $W^{(k)}_{n,\tau}$ is given by

$$\frac{\partial \mathcal{L}_{\text{tot}}}{\partial W^{(k)}_{n,\tau}} = \frac{\partial \mathcal{L}_{\text{KL}}}{\partial W^{(k)}_{n,\tau}} = \sum_t \frac{\partial \mathcal{L}_{\text{KL}}}{\partial \hat{X}_{n,t}} \frac{\partial \hat{X}_{n,t}}{\partial W^{(k)}_{n,\tau}}. \tag{A.13}$$

Calculating $\frac{\partial \mathcal{L}_{\text{KL}}}{\partial \hat{X}_{n,t}}$ from Equation (3.3) gives

$$\frac{\partial \mathcal{L}_{\text{KL}}}{\partial \hat{X}_{n,t}} = 1 - \frac{X_{n,t}}{\hat{X}_{n,t}}. \tag{A.14}$$

Calculating $\frac{\partial \hat{X}_{n,t}}{\partial W_{n,\tau}^{(k)}}$ from Equation (3.2) gives

$$\frac{\partial \hat{X}_{n,t}}{\partial W_{n,\tau}^{(k)}} = \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right). \tag{A.15}$$

Substituting Equations (A.14) and (A.15) in Equation (A.13) gives

$$\frac{\partial \mathcal{L}_{\text{tot}}}{\partial W_{n,\tau}^{(k)}} = \frac{\partial \mathcal{L}_{\text{KL}}}{\partial W_{n,\tau}^{(k)}} = \sum_t \left(1 - \frac{X_{n,t}}{\hat{X}_{n,t}}\right) \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right), \tag{A.16}$$

so that

$$\frac{\partial \mathcal{L}_{\text{tot}}^{+}}{\partial W_{n,\tau}^{(k)}} = \sum_t \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right), \tag{A.17}$$

$$\frac{\partial \mathcal{L}_{\text{tot}}^{-}}{\partial W_{n,\tau}^{(k)}} = \sum_t \frac{X_{n,t}}{\hat{X}_{n,t}} \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right). \tag{A.18}$$

Substituting Equations (A.17) and (A.18) in Equation (A.12) gives the update rule for $W_{n,\tau}^{(k)}$:

$$W_{n,\tau}^{(k)} \leftarrow W_{n,\tau}^{(k)} \frac{\sum_t \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right) \left(\frac{X_{n,t}}{\hat{X}_{n,t}}\right)}{\sum_t \sigma\left(a_k\right) \sigma\left(G_{k,t-\tau}\right)}. \tag{A.19}$$

A.3 Additive Gradient-Descent Update for a

$\mathcal{L}_{\text{tot}}$ is minimized with respect to a using gradient-descent, see Equation (3.12). The partial derivative $\frac{\partial \mathcal{L}_{\text{KL}}}{\partial a_k}$ given by

$$\frac{\partial \mathcal{L}_{\text{KL}}}{\partial a_k} = \sum_{n,t} \frac{\partial \mathcal{L}_{\text{KL}}}{\partial \hat{X}_{n,t}} \frac{\partial \hat{X}_{n,t}}{\partial a_k}, \tag{A.20}$$

$$\text{with } \frac{\partial \hat{X}_{n,t}}{\partial a_k} = \sum_\tau \sigma\left(a_k\right) \sigma\left(-a_k\right) \sigma\left(G_{k,t-\tau}\right) W_{n,\tau}^{(k)}. \tag{A.21}$$

Substituting Equations (A.14) and (A.21) in Equation (A.20) and rearranging gives the following expression for the derivative of $\mathcal{L}_{\text{tot}}$ with respect to a_k:

$$\frac{\partial \mathcal{L}_{\text{tot}}}{\partial a_k} = \frac{\partial \mathcal{L}_{\text{KL}}}{\partial a_k} = \sigma(-a_k) \sum_{n,t} \left(1 - \frac{X_{n,t}}{\hat{X}_{n,t}}\right) \hat{X}^{(k)}_{n,t}, \tag{A.22}$$

$$\text{with } \hat{X}^{(k)}_{n,t} = \sum_{\tau} \sigma(a_k)\sigma(G_{k,t-\tau}) W^{(k)}_{n,\tau}. \tag{A.23}$$

References

[1] Mikkel N. Schmidt and Morten Mørup. "Nonnegative matrix factor 2-D deconvolution for blind single channel source separation". In: *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2006), Charleston, SC, USA* (2006), pp. 700–707. ISSN: 03029743. DOI: 10.1007/11679363_87.

[2] Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization". In: *Advances in Neural Information Processing Systems* (2001), pp. 556–562.

B

Tracklists of the dataset for the automated DJ system

Table B.1: Tracklist of training set (117 songs)

A-Cray	Stomper
Addergebroed	Shadows (Culprate Remix)
AKOV	Heavyweight
AKOV	Construct
AKOV	Crude Tactics
AKOV	Omega
Allied	Godspeed
Apex	Inner Space
AZEDIA	Something (Keeno Remix)
Bensley	Tiptoe
Black Sun Empire & State of Mind	Ego

Continued on next page...

Table B.1 – *Continued from previous page*

Black Sun Empire, State Of Mind feat PNC	Until The World Ends
Blu Mar Ten	Whisper (feat. Kirsty Hawkshaw)
Boosta & Atmos T	The Things You Do
Calibre & High Contrast	Mr. Majestic
Camo & Krooked	Afterlife
Camo & Krooked	Climax
Cause4Concern	Headroom (Audio VIP)
Colossus	Riptide
Commix	Painted Smile
Commix	Be True
Culture Shock	City Lights
Culture Shock & Brookes Brothers	Rework
D-Bridge	True Romance VIP
Dementia, Rregula & Disphonia	Doom Loop (Mefjus Remix)
Disphonia	War Bunker
Disphonia & Camelorg	Commit (Myselor Remix)
Droptek	The Covenant
DRS & LSB	The View (feat. Tyler Daley)
Ed Rush & Optical	Bacteria
Ed Rush & Optical	Chubrub
Ed Rush & Optical	Snaggletooth
Ed Rush & Optical feat Rymetyme	City 17
Emperor	Monolith VIP
Emperor & Mefjus	Void Main Void
Emperor & Mefjus	Disrupted
Fourward	Side by Side
Fourward	Stressed Out
Fourward	Talk to Me
Fourward Ft. Kyza	Countdown

Continued on next page...

Table B.1 – *Continued from previous page*

Fred V & Grafix	Games People Play
Fred V & Grafix	Bladerunner
Fred V & Grafix	Recognise
Hybrid Minds	Lost
Hybrid Minds	Meant To Be
Hybris	eWaste
HYQXYZ	Repeaters
Indigo	Absent
InsideInfo & Mefjus	Blunt
InsideInfo & Mefjus	Mythos
InsideInfo & Mefjus	Footpath (feat. The Upbeats)
Joe Ford	Behemoth
Joe Ford	Stride
Keeno	Borderless (feat Whiney)
London Elektricity	The Plan That Cannot Fail (Logistics Remix)
Mark Knight, Skin	Nothing Matters (Noisia Remix)
Mefjus	Blitz
Mefjus	Far Too Close
Mefjus	Signalz
Mefjus feat. Zoe Klinck	Blame You
Metrik	Nightdrive
Mind Vortex	Colours
Mind Vortex	Underworld
Neosignal	Sequenz (Mefjus Remix)
Noisia	Soul Purge Ft Foreign Beggars
Noisia	Diplodocus
Noisia	Could This Be
Noisia	Shellshock Ft Foreign Beggars
Noisia & Phace	Stagger
Noisia & TeeBee	Moon Palace
Noisia ft. Giovanca	My World
Phace & Noisia	Micro Organism

Continued on next page...

Table B.1 – *Continued from previous page*

Phase	Time Flies
Rawtekk	Sprouted & Formed
Rawtekk	No More Vaccine
RUFUS	Desert Night (Hybrid Minds Remix)
S.P.Y	Manicured Reality (feat. LowQui)
S.P.Y	Riding The Void
S.P.Y	Step & Flow
S.P.Y	Frozen (feat. Diane Charlemagne)
S.P.Y & BCee	Stardust
S.P.Y & Total Science	Double Trouble
S.P.Y & Total Science	Guidance
S.P.Y feat. Diane Charlemagne	Dusty Fingers
S.P.Y feat. DRS	Stand Alone
S.P.Y feat. Suku Of Ward 21	Rise Again
Spor	Coconut
Spor	Know You
Spor	As I Need You
Spor	Figaro
Spor	Halogen
Spor	Kingdom
Spor	Blurred Vision
Spor	Empire
Spor & Phace	Woodruff
Spor, Icicle & Linguistics	Mind Of An Insomniac
Stan SB	Anyone Out There
State Of Mind	No-Operative
State Of Mind	Bigger Faster Stronger
State Of Mind	Sunking
State Of Mind	Where You At
State Of Mind, Black Sun Empire	Unconscious

Continued on next page...

Table B.1 – *Continued from previous page*

Teddy Killerz	New Drums
Teknian, Disprove & Ordure	Lockheed
The Clamps	Dvouglas
The Clamps	Schoolchild
The Prodigy	Nasty (Spor Remix)
Ulterior Motive	Sideways
Ulterior Motive	Tape Pack
Ulterior Motive & Hybris	Bring Out
Wilkinson	Redemption
Wilkinson	Take You Higher
Wilkinson	Sleepless
Wilkinson feat. Iman	Need To Know
Wilkinson featuring P Money and Arlissa	Heartbeat
Wilkinson featuring Tom Cane	Half Light
Zombie Cats & Mefjus	Must Eat

Table B.2: Tracklist of test set (43 songs)

Ble3k	Breach
Bredren	Nina Dobrev
Calyx & TeeBee	Long Gone
Calyx & TeeBee	Skank
Camo & Krooked	Hot Pursuit (Indivision & Cosmic Remix)
Camo & Krooked	In the Future (feat. Jenna G & Futurebound)
Camo & Krooked	Menace (Mefjus Remix)
Camo & Krooked	Run Riot
Chase & Status	No Problem
Cyantific	Colour in the Shadows (feat. Benji)
DC Breaks	Swag

Continued on next page...

Table B.2 – *Continued from previous page*

Dimension	Whip Slap
Dimension	Pull Me Under (ft. Raphaella)
Distorted Minds & DJ Hazard	Mr Happy
DLR & Break	Human Error
Emperor	Precursor
Hectix	The Return
Icicle	Dreadnaught ft. SP:MC
InsideInfo & Smooth	Hear Me Roar
June Miller	Chain Of Strength
MaxNRG	Hide Away My Heart
Mefjus & InsideInfo	Repentance
Neonlight	Heavy Bettie
Netsky	Rise and Shine
Netsky	Tomorrow's Another Day (VIP Mix)
Netsky	Love Has Gone
Netsky	Memory Lane
Netsky	Everyday
Netsky	Give & Take
Netsky	Black & Blue
Netsky	Iron Heart
Netsky	Escape
Netsky	Come Alive
Noisia	Stigma
Noisia	Hubcap
Noisia & Calyx & TeeBee	Hyenas
Noisia & The Upbeats	Omnivore
Noisia & The Upbeats	Mouthbreather
Noisia & The Upbeats	Dustup
Noisia & The Upbeats	Dead Limit
Skorp	High Pressure
Smooth	Drone
The Clamps	Nerves

Table B.3: Tracklist of additional test set (220 songs)

AKOV	Rubix
AKOV	Negative Space
Andy C	Haunting
Artifact	Unleashed
Artifact	Dystopia
Audio	Collision
Black Sun Empire	Breach (The Upbeats Remix)
Black Sun Empire	Potemkin (S.P.Y Remix)
Black Sun Empire	B'Negative (Phace & Misanthrop Remix)
Black Sun Empire	Arrakis
Black Sun Empire	Don't You (State of Mind Remix)
Black Sun Empire	Arrakis (Noisia Remix)
Black Sun Empire	All Is Lost (Memtrix Remix)
Black Sun Empire	The Rat (Gridlok Remix)
Black Sun Empire	Delorean
Black Sun Empire	All Is Lost
Black Sun Empire	Dawn Of A Dark Day
Black Sun Empire	Shuffle
Black Sun Empire	Sideways feat. Illy Emcee (Optiv & BTK Remix)
Black Sun Empire	Killing the Light feat. Inne Eysermans (Icicle Remix)
Black Sun Empire	Extraction (Rido Remix)
Black Sun Empire	Transmission (Mindscape Remix)
Black Sun Empire	Brainfreeze (Neonlight Remix V2)
Black Sun Empire	Dawn of a Dark Day feat. Foreign Beggars (Prolix Remix)
Black Sun Empire	Chaingang (Jade Remix)
Black Sun Empire	Monologue (Ulterior Motive Remix)

Continued on next page...

Table B.3 – *Continued from previous page*

Black Sun Empire	Bitemark (Zardonic Remix)
Black Sun Empire	Eraser
Black Sun Empire & Audio	Drizzle
Black Sun Empire & Jade	Deadhouse (InsideInfo & Mefjus Remix)
Black Sun Empire & Noisia	Lead Us (Audio Remix)
Black Sun Empire & Noisia	Feed The Machine
Black Sun Empire & State Of Mind	Long Time Dead
Break & Mako & Fields & Villem	Shadowlines
Brookes Brothers	Carry Me On ft Chrom3 (Club Mix)
Calibre	Even if
Calyx & TeeBee	A Day That Never Comes
Calyx & TeeBee	Where We Go Feat. Doctor
Calyx & TeeBee	Make Your Choice
Calyx & TeeBee	Confession
Calyx & TeeBee	Sawn Off
Chase & Status	Hurt You
Chris.Su	Datahub (Mindscape Remix)
Chris.Su	Illusions
Commix	Satellite Type 2
Culture Shock	Troglodyte
Culture Shock	Raindrops
Culture Shock	Low Frequency
Culture Shock	I Remember
Cyantific	Ice Cream (Vanilla Mix)
David Zowie	House Every Weekend (Loadstar Remix)
DC Breaks	Swag (Audio Remix)
DC Breaks	Gambino
Delta Heavy	White Flag (VIP)
Dementia, Disphonia & Rregula	High Times

Continued on next page...

Table B.3 – *Continued from previous page*

Dimension	Love To Me
Dimension	Digital World VIP
Dimension	Move Faster
Dimension	Crowd Reaction VIP
Disphonia	Headfirst
DJ Fresh	Hot Right Now (Camo & Krooked Remix)
DJ Marky & S.P.Y	Yellow Shoes
Doctrine	Airlock
Dodge & Fuski vs Culprate	Vice (Phetsta Remix)
Drumsound & Bassline Smith	Daylight
Drumsound & Bassline Smith	Phantasm
Drumsound & Bassline Smith	Close
Dub Phizix & Skeptical	Marka feat. Strategy
Dub Phizix & Strategy	Bounce
Dvbbs & Borgeous	Tsunami (Friction Remix)
Ed Rush & Optical	Long Stay (feat. Ryme Tyme)
Ed Rush & Optical	Pacman (Ram Trilogy Remix)
Etherwood	Souvenirs (ft. Zara Kershaw)
Fourward	Streetknowledge
Fourward	Spike
Fourward	Black Tooth Grin
Frankee & Caan	Deep down
Fred V & Grafix	Maverick Souls
Fred V & Grafix feat. Reija Lee	Just A Thought
Friction & Skream Ft. Scrufizzer, P Money & Riko Dan	Kingpin (Calyx & TeeBee Remix)
Friction vs Camo & Krooked	Stand Up (feat. Dynamite MC) (Sigma Remix)
Future Prophecies	September (feat. Roger Ludvigsen)
Hive	Neo (Audio Remix)
Hybrid Minds	Meant To Be (InsideInfo Remix)

Continued on next page...

Table B.3 – *Continued from previous page*

Hybrid Minds	Summer Rain
Hybrid Minds	I'm Through
Indiana	Mess Around (Etherwood Remix)
Jakwob	Blinding (Hybrid Minds Remix)
Jakwob ft. Maiday	Fade (Keeno Remix)
James Marvel	Way Of The Warrior (feat. MC Mota)
John B	The Journey (feat CODE 64 - Metrik instrumental Remix)
June Miller	Robots & Romans - Audio Remix
K-Tee	Hypnotize VIP
KOAN Sound	Forgotten Myths
Kove	Hurts (feat. Moko) (LSB Remix)
Kove	Searching
Krakota	In the Area (feat. Lifford)
Krakota	Lazy Bones
Legion, Logam & NC-17	Gate
Loadstar	Bomber
Loadstar	Black and White (Ft. Benny Banks) (Hamilton Remix)
Logistics	Together
Logistics	Cosmonaut
London Elektricity	Meteorites (feat.Elsa Esmeralda) (Danny Byrd Remix)
London Grammar	Hey Now (Keeno Remix)
Lynx	Clap Track
Macklemore & Ryan Lewis	Thrift Shop (Maduk Remix)
Maduk	Voyager
Maduk	Maduk ft Veela - Ghost Assasin (Hourglass Bonusmix)
Maduk	Feel Good
Matrix & Futurebound (feat. Luke Bingham)	All I Know (Matrix & Futurebound Rolling Out DJ Mix)

Continued on next page...

Table B.3 – *Continued from previous page*

Matrix & Futurebound	Skyscraper
Maverick Sabre	I Used To Have It All (Delta Heavy Remix)
Mefjus	Contemporary
Mefjus	Dissuade
Mefjus ft. Icicle	Leakproof
Memtrix	Capsize
Memtrix	Mind Control
Memtrix	Ethereal
Memtrix	Limelight
Memtrix	Slipper
Metrik	I See You (feat. Kathy Brown)
Metrik	Slipstream
Mind Vortex	Gravity
Mind Vortex	Till the Sun Comes
Mob Tactics	Lifer
Mob Tactics	Roadhouse
Muffler	Hear Me Scream
Murdock	Black Out VIP
Neonlight & Wintermute	Influx
Netsky	The Lotus Symphony
Netsky	Rio
Netsky	We Can Only Live Today (Puppy) (feat. Billie) (Camo & Krooked Remix)
Netsky	I Refuse
Netsky	Come Back Home
Netsky	Lost In This World
Nu:Logic	Morning Light
Optiv	Krakpot
Optiv & BTK	Journey From The Light
Optiv & BTK	Hack and Slash
Optiv & BTK	Drop the Funk

Continued on next page...

Table B.3 – *Continued from previous page*

Optiv & BTK	Get Dark
Optiv & BTK	Whatever (Mefjus Remix)
Optiv & BTK	Dead Beat
Optiv & BTK	You Got Me So
Optiv & CZA	Sabretooth (Audio Remix)
Original Sin	Therapy VIP
Original Sin	Back For More
Original Sin and Taxman	Casino
Phace	Open Your Eyes
Phase	Return The Flavour
Phase	Eternal Truth
Phase & Bredren	Moments
Prolix	Pick Pocket
QO	Evil Dead
Rawtekk	To Be A Space Monkey
Rene LaVice	Air Force 1
Riya	Fear Bites (feat. Dynamite MC, Villem, McLeod)
Rockwell	Detroit
S.P.Y	By Your Side (Logistics Remix)
ShockOne	Infinity's Silence
Sigma	Rudeboy
Skeptical	Imperial
Smooth	Revenge
SpectraSoul	Bugsy
SpectraSoul feat. Tamara Blessa	Away With Me (Calibre Remix)
Spor	Bad Company UK - Bullet Time (Spor Remix)
Spor	Overdue (feat. Tasha Baxter)
Spor	Aztec
Spor	Do Not Shake
Spor	Always Right Never Left
Spor	The Hole Where Your House Was

Continued on next page...

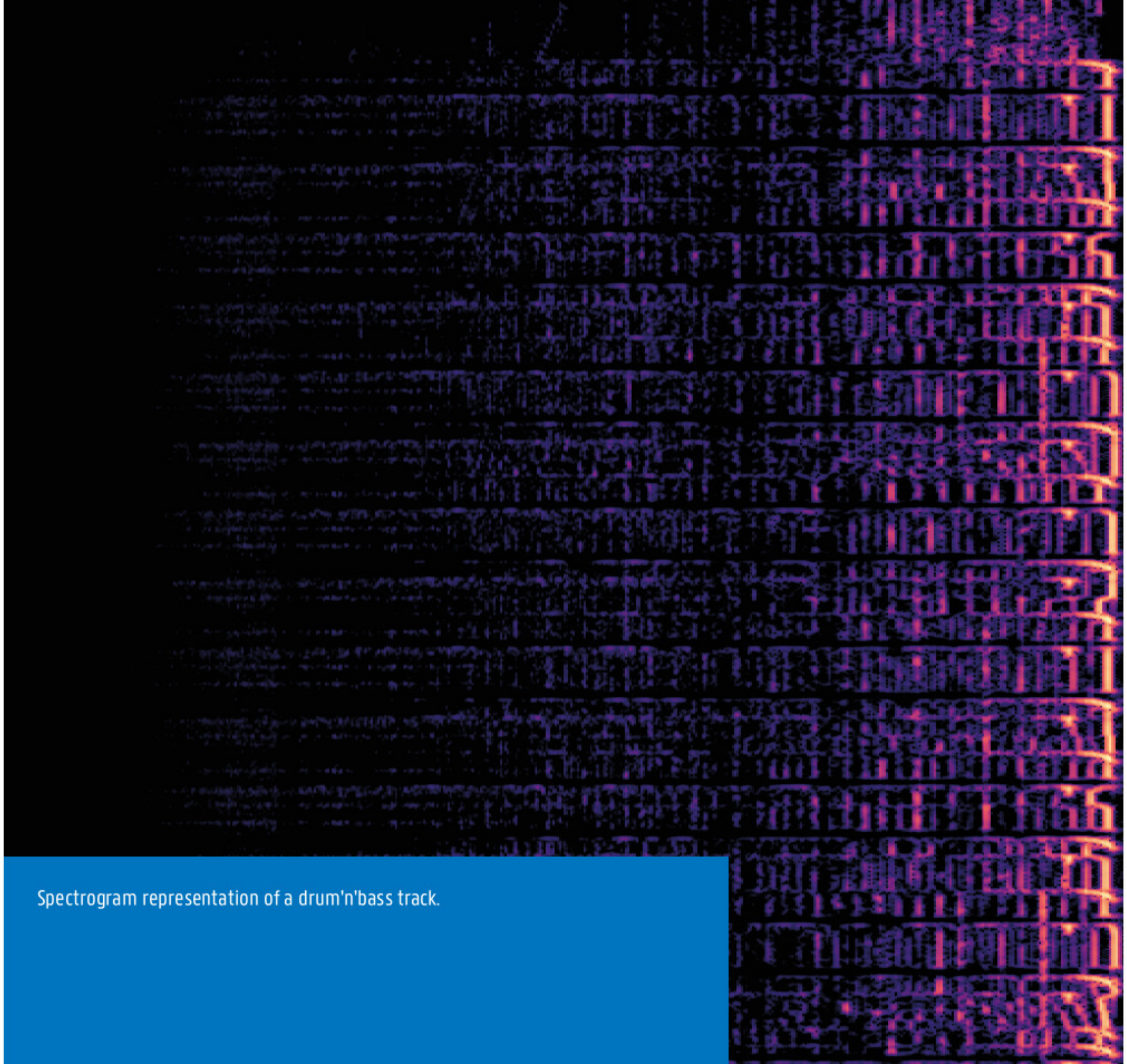
Table B.3 – *Continued from previous page*

Spor	Blueroom
Spor	Full Colour
Spor	If You Cry
Spor	103 Degrees
Spor	Arms House
Spor	Strange Heart
State Of Mind	Real McCoy
State Of Mind	Danse Macabre
State Of Mind	Snakechamber
State Of Mind	Dark Man
State Of Mind	Mindslicer
State Of Mind	Fast Life
State Of Mind	Helios
State Of Mind	Flash point
State Of Mind	Back to the jungle
State Of Mind	U Control Me
State Of Mind	Paint
State Of Mind	Barricade
State Of Mind	Afterlife
State Of Mind	Rain Maker
State Of Mind	Response Signal
State Of Mind	Dune
State Of Mind, MC Dino	Ghosts
State Of Mind, Nymfo	Put It On
State Of Mind, Percieve	Mr. Cover Up
State Of Mind, Sacha Vee	Black Raven
Sub Focus	Flamenco
Sub Focus	Triple X
Sub Focus	Out
Sub Focus	Timewarp
Sub Focus	Eclipse
Sub Focus	Airplane

Continued on next page...

Table B.3 – *Continued from previous page*

Sub Focus	Endorphins (Sub Focus vs Fred V & Grafix Remix)
T & Sugah	Deep Space
Tantrum Desire	Unreal (feat. Ayve)
TC	Tap Ho
Technimatic	Transference
The Clamps	Strains
The Prodigy	Voodoo People (Pendulum Remix)
The Prototypes	Lights
The Prototypes	Habitation
The Upbeats	Babylon (feat. Orifice Vulgatron)
The Upbeats	Say Go (feat. Mara TK)
The Upbeats	Mediums
The Upbeats	Alone feat. Tasha Baxter (Fourward Remix)
Trei & State of Mind	Breed
Volatile Cycle	Horizon
Zhu	Faded (Delta Heavy Bootleg)



Spectrogram representation of a drum'n'bass track.