Universitat Pompeu Fabra
Barcelona

Automatic Characterization and Generation of Music Loops and Instrument Samples for Electronic Music Production

António Ramires

TESI DOCTORAL UPF / 2022

Thesis Directors:

Dr. Xavier Serra i Casals
Music Technology Group
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

Dr. Frederic Font Corbera
Music Technology Group
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

Music Technology Group (http://mtg.upf.edu), Department of Information and Communication Technologies (http://upf.edu/dtic), Universitat Pompeu Fabra (http://upf.edu), Barcelona, Spain.

This thesis has been carried out at the Music Technology Group (MTG) of Universitat Pompeu Fabra in Barcelona, Spain, from September. 2018 to October. 2022. It is supervised by Dr. Xavier Serra i Casals and Dr. Frederic Font Corbera. This research has been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

Acknowledgments

I would like to start by expressing my most immense gratitude to Frederic Font and Xavier Serra for the guidance and supervision they provided me through these years. To Frederic for all the music and music technology discussions, for teaching me how to code properly, for showing me how beautiful Freesound is, for all the work reviewing and guidance. To Xavier for teaching me how to think strategically, for all the knowledge that only someone who knows this field so well can transmit and, above all, for giving me this opportunity to work in the MTG and MIP-Frontiers. I still remember what I thought would be just one more interview in the selection process, but it was a Skype meeting to let me know I would be able to pursue a dream PhD in helping electronic musicians through technology.

A very special thanks to my first mentor and forever friend, Matthew Davies. He is one of the kindest and most dedicated persons I have ever met and the person responsible for showing me MIR, making me fall in love with it and providing me with all the tools and knowledge to be able to start this PhD journey.

My deepest love and thankfulness to Catarina, for making the move to Barcelona a smooth ride, for all the plane travels at inappropriate times she took to make Barcelona feel like my home, and for all the support and motivation she gave me to take one of the biggest steps of my life. For all the love, kindness, patience and emotional support she provided me with all these years, which were essential to this PhD.

To my whole family, but especially to my parents Ana Paula and António, my tia Helena, Mimi and Zizi, who have provided me with their own guidance and pressure to wrap up this thesis, but most of all for the love, support and comfort. To my grandparents and to tia Lulu, my aunts, uncles and cousins, who gave me renewed energy through relaxing times and funny conversations.

To all the people I have met during these years in Barcelona. To all my MTG friends for all the feedback, great discussions and, above all, great times and fun breaks: Alastair, Albin, Alia, Andres, Angel, Behzad, Benno, Dmitry, Eduardo, Enric, Furkan, Guillem, Helena, Jordi, Jorge, Juan, Jyoti, Lorenzo, Luis, Marius, Miguel, Minz, Olga, both Pablos, Perfe, Philip, Pritish, Sergi, Sergio, Thomas, Vsevold, Rafa and both Xaviers. To Cristina, Sonia and Lydia for all the administrative help and patience towards such an unorganized person as myself. To all the MIP-Frontiers gang. To all the researchers I have met along the way, especially the ones I worked with: Jake, Patricio, Gilberto, Gerard, Jason, Jordan, Julian, Yi-Hsuan and Joann. To the Jokkoo family for all the musical inspiration, fun times and crazy nights in Barcelona: Maguette, Baba, Mookie, B4mba, Opoku, Mans O, Ikram, Silva, Mickel and Debruit. To the crew at Beatoven.ai for believing in my work and providing me with time and understanding to finish this thesis, it's a pleasure working with you all.

Last but certainly not least, to all my Coimbra friends whose friendship and support have been there for so many years and which keeps growing each day, with a special thanks to Bicá, Brás, Cabrita, Chico, Craveiro, Diogo, Fábio, Filipe, Gonçalo, Inês, João André, Mariana, Sandro and Silveira.

Abstract

Repurposing audio material to create new music - also known as sampling - was a foundation of electronic music and is a fundamental component of this practice. Loops are audio excerpts, usually of short duration, that can be played repeatedly in a seamless manner. These loops can serve as the basis for songs that music makers can combine, cut and rearrange and have been extensively used in Electronic Dance Music (EDM) tracks. Similarly, the so-called "one-shot sounds" are smaller musical constructs that are not meant to be looped but are also typically used in EDM production. These might be sound effects, drum sounds or even melodic phrases. Both loops and one-shot sounds have been available for amateur and professional music makers since the early ages of electronic music. Currently, large-scale databases of audio offer huge collections of audio material for users to work with. The navigation on these databases is, however, still heavily focused on hierarchical tree directories. Consequently, sound retrieval is tiresome and often identified as an undesired interruption in the creative process.

In our work, we address two fundamental methods for navigating sounds: characterization and generation. Characterizing loops and one-shots in terms of their instruments or instrumentation (e.g. drums, harmony, melody) allows for organizing unstructured collections and a faster retrieval for music-making. Generation of loops and one-shot sounds enables the creation of new sounds not present in an audio collection through interpolation or modification of the existing material. To achieve this, we employ deep-learning-based data-driven methodologies for classification and generation.

We start by applying convolutional neural networks to the task of instrument classification with a large-scale dataset of synthesized sounds augmented with audio effects, achieving high accuracy. Then, we present a large annotated collection of musical loops from Freesound, along with several applications and use cases, which enable further research in loop characterization. Using this dataset, we present an algorithm for classifying the instrumentation role a loop can take in a music composition and show that it can be applied to finding musical structure.

The first contribution to generation is a neural synthesizer of percussive sounds based on high-level semantic concepts, along with a dataset of percussive one-shots from Freesound. We then extend this architecture to generate drum loops and evaluate several loss functions in terms of audio quality. Finally, we employ generative adversarial networks to create drum sounds with significantly higher audio quality and present the results of a user study to understand the preference for synthesis controls.

Developments in music technology have revolutionized music creation and have enabled a vast amount of new music genres. Tools like the ones we propose have the potential capacity to change the way music is made, perhaps in a similar way to how sampling revolutionized music in the past.

Resum

La reutilització del material d'àudio per crear música nova -també coneguda com a mostreig- va ser una base de la música electrònica i és un component fonamental d'aquesta pràctica. Els "loops" són fragments d'àudio, generalment de curta durada, que es poden reproduir repetidament. Aquests "loops" poden servir com a base per a cançons que els creadors de música poden combinar, retallar i reordenar i s'han utilitzat àmpliament a les pistes de Electronic Dance Music (EDM). De la mateixa manera, els anomenats "one-shot sounds" són construccions musicals més petites que no estan destinades a ser reproduïdes en bucle, però que també s'utilitzen en la producció d'EDM. Aquests poden ser efectes de so, sons de bateria o fins i tot frases melòdiques. Tant els "loops" com els "one-shot sounds" han estat disponibles per als creadors de música aficionats i professionals des del principi de la música electrònica. Actualment, les bases de dades d'àudio a gran escala ofereixen col · leccions enormes de material d'àudio perquè els usuaris puguin treballar. La navegació per aquestes bases de dades, però, encara està molt centrada en directoris d'arbre jeràrquic. Cosa que fa que la recuperació del so sigui tediosa i sovint s'identifica com una interrupció no desitjada en el procés creatiu.

Al llarg d'aquesta tesi, tractem dos mètodes fonamentals per navegar els sons: la caracterització i la generació. Caracteritzar "loops" i "one-shot sounds" pel que fa als seus instruments o instrumentació (per exemple, bateria, harmonia, melodia) permet organitzar col · leccions no estructurades d'audio i la cerca més ràpida d'àudio per a la creació musical. La generació "loops" i "one-shot sounds" permet la creació de nous sons no presents en una col · lecció d'àudio mitjançant la interpolació o modificació del material existent. Per aconseguir-ho, utilitzem metodologies basades en xarxes neuronals per a la classificació i la generació.

Comencem aplicant xarxes neuronals convolucionals a la tasca de classificació d'instruments amb un conjunt gran de dades amb sons augmentats utilitzant efectes d'àudio, aconseguint una gran precisió. A continuació, presentem una gran col · lecció anotada de "loops" musicals de Freesound, juntament amb diverses aplicacions i casos d'ús, que permeten més recerca en la caracterització de bucles. Utilitzant aquest conjunt de dades, presentem un algorisme per classificar el paper instrumental que pot tenir en un "loop" en una composició musical i mostrem que es pot utilitzar per a descriure l'estructura musical de la peça.

La primera contribució respecte a la generació de sons és un sintetitzador neuronal de sons percussius basat en conceptes semàntics d'alt nivell, juntament amb un conjunt de dades de "one-shot sounds" percussius de Freesound. A continuació, extenem aquesta arquitectura per generar "loops" de bateria i evaluem diverses funcions de cost en termes de qualitat d'àudio. Finalment, utilitzem xarxes adversàries generatives per crear sons de bateria amb una qualitat d'àudio significativament més alta i presentem els resultats d'un estudi perceptual per entendre les preferències dels usuaris respecte els controls de síntesi.

Els desenvolupaments de la tecnologia musical han revolucionat la creació musical i han permès una gran quantitat de nous gèneres musicals. Eines com les que proposem tenen la capacitat potencial de canviar la manera com es fa la música, potser d'una manera similar a com el mostreig va revolucionar la música en el passat.

Resumen

Reutilizar material de audio para crear música nueva - también conocido como sampling - fue un cimiento de la música electrónica y un componente fundamental de su práctica. Los bucles son extractos de audio, normalmente de corta duración, que pueden ser reproducidos repetidamente y de forma ininterrumpida. Estos bucles pueden servir como base para canciones que los creadores de música pueden combinar, cortar, y reorganizar, y han sido ampliamente utilizados en pistas de Electronic Dance Music (EDM). De la misma manera, los llamados "sonidos one-shot" son constructos musicales más cortos cuya intención no es ser reproducidos en bucle y también son frecuentemente usados en la producción de música EDM. Estos pueden ser efectos de sonido, sonidos de batería, o incluso frases melódicas. Tanto los bucles como los sonidos one-shot, han estado disponibles para creadores de música aficionados y profesionales desde los comienzos de la música electrónica. Actualmente, las bases de datos de audio a gran escala ofrecen grandes colecciones de material de audio para que los usuarios utilicen. Sin embargo, la navegación por esas bases de datos todavía se basa principalmente en jerarquías de árboles de carpetas. Consecuentemente, la recuperación de sonidos es tediosa y a veces se identifica como una interrupción no deseada del proceso creativo.

En nuestro trabajo, abordamos dos métodos fundamentales para la navegación de sonidos: caracterización y generación. Caracterizar bucles y one-shots en términos de sus instrumentos y su instrumentación (e.g. baterías, armonía, melodía) permite organizar colecciones no estructuradas y una recuperación más rápida para la creación de música. La generación permite la creación de nuevos sonidos no presentes en una colección de audio a través de la interpolación o modificación del material existente. Para lograr esto, empleamos metodologías basadas en datos y deep learning para la clasificación y generación. Comenzamos aplicando redes neuronales convolucionales a la tarea de la clasificación de instrumentos con un dataset a gran escala de sonidos sintetizados aumentados con efectos de audio, consiguiendo gran exactitud. Luego, presentamos una gran colección anotada de bucles musicales de Freesound, junto con múltiples aplicaciones y casos de uso, los cuales permiten más investigación respecto a la caracterización de bucles. Usando esta colección, presentamos un algoritmo para clasificar el rol de la instrumentación que un bucle puede tomar en una composición musical, y mostramos que puede ser usado para encontrar la estructura musical.

La primera contribución a la generación es un sintetizador neuronal de sonidos percusivos basados en conceptos semánticos de alto nivel, junto con un dataset de sonidos percusivos one-shot de Freesound. Después ampliamos esta arquitectura para generar bucles de batería y evaluar varias funciones de pérdida en términos de calidad de audio. Finalmente, empleamos redes generativas adversarias para crear sonidos de batería con significativamente más calidad de audio y presentamos los resultados de un estudio con usuarios para entender las preferencias para los controles de la síntesis.

Los desarrollos en la tecnología musical han revolucionado la creación musical y han permitido la aparición de una gran cantidad de nuevos géneros musicales. Herramientas como las que proponemos tienen la capacidad de cambiar la forma en la que se hace música, quizás de un modo similar a como el sampleo revolucionó la música en el pasado.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

The democratization of Electronic Music Production (EMP) is bringing advanced music creation tools to anyone who is interested in making music. The developments in music technology, especially the development of Digital Audio Workstation (DAW) and virtual instruments, have led to a democratization of music production. Music makers, both professional and amateur, have the tools for composing, playing instruments and processing audio all on their personal computers. The technological evolution paved the way for making EMP accessible to anyone who owns a personal computer and, with this, we see amazing pieces of music surfacing from around the world from artists who have never had formal music training.

Repurposing audio material to create new music, also known as sampling, was a foundation of Electronic Music (EM) and is a fundamental component of this practice. One-shot sounds — short isolated single instrumental notes, chords or sound effects — and loops — repeating patterns associated with a particular instrument — form the basis of Electronic Dance Music (EDM) tracks (Butler, 2006; Ratcliffe, 2014). Drum loops such as the 'Funky Drummer' and the 'Amen Break'[1], originally from Funk records, have been sampled literally thousands of times and are fundamental to the timbral character of hip-hop and many EM genres.

While in early hip-hop these audio excerpts were collected from existent music records, the democratization of music creation together with technological developments led to novel ways of distributing audio material. As an alternative to using samples from copyrighted music, royalty-free sample collections aimed directly at music makers surfaced. The way these collections were distributed was linked with the technology available at the time, from sample CDs in the 1990s, to small-sized sample packs distributed over the internet, to the

[1] An overview of how this loop reached mainstream music is presented here: https://bbc.com/news/magazine-32087287

large-scale collections made available with modern cloud storage technologies. Together with the increase in the amount of material available for users, the difficulty in navigating these sound collections also rose. This thesis presents work towards improving the navigation of sound collections, through the high level characterization and the generation of sounds similar to the sounds inside these collections.

In this chapter, we will review the history of sampling practices in EM, understand some of the issues identified by music makers when navigating large-scale collections of sampleable material and introduce the methodologies presented in this thesis towards enhancing this navigation.

1.1 Context

> "An organ with each key linked to a turntable that would have appropriate discs put on it as required; let's suppose that the keyboard of this organ switches on the record players simultaneously or one after the other (...). In theory we get a mother instrument, capable of replacing not only all existing instruments but every conceivable instrument." Schaeffer (1952)

In these notes from the 23rd of April, 1948, Pierre Schaeffer envisioned the concept of an instrument that would later create a revolution on the music making paradigm, the sampler. The field of EM has often been led by composers and inventors with a need to invent a way to realize their musical visions (Holmes, 2012). One of these inventors/composers was Pierre Schaeffer. In order to better understand EMP practices and evolution, we will review the history of sound in EM. The context section will, therefore, focus on the evolution of the use of recorded audio in EM. We will start by reviewing Schaeffer's work, followed by the work of tape composers, arriving then at the possibilities provided by digital samplers and computers. We will finish by analysing large sound databases and licenses for remixing copyrighted pieces of work.

1.1.1 The Use of Recorded Audio for Composition in Electronic Music

The use of sound and noise in the process of music creation, together with electronic signals was pioneered by the work of Pierre Schaeffer with Pierre Henry. The two collaborators were responsible for beginning of a new era of EM, that of the recorded sound (Holmes, 2012). This new era was able to withdraw the dependency on performance in EM creation, allowing the establishment of music studios around the world.

During World War II, at the time of the German occupation of France, Schaeffer worked for the Radiodiffusion Télévision Françaises (RTF), where he founded the Studio d'Essai in 1942. Due to his work in RTF, Schaeffer had access to a variety of equipment for radio broadcasting, which he was able to employ in his experiments and in the production of his musical pieces. These included turntables, mixers, microphones, a direct-to-disc cutting lathe, which allowed him to record audio to acetate discs, and a library of sound effects recordings which were used for radio production.

During this research, Schaeffer focused his attention on using recording techniques for isolating naturally produced sound events (Manning, 2004) and, in 1948, started to investigate how these sounds could be used as a basis for music production. These experiments led to the creation of *"Études de bruits"* (Studies of noises), a series of short studies which included one of Schaeffer's most famous works, *"Étude aux chemins de fer"* (Railway Study). This composition challenged Schaeffer to musically organize recordings of sound produced by six locomotives at the Batignolles station in Paris (Palombini, 1993), which included the sounds of the locomotive's whistling, trains accelerating and wagons passing over joints in the rails (Manning, 2004). During this work, Schaeffer discovered some techniques for editing the recording material that are still used in today's EM, such as glueing sounds together by playing and re-recording them, playing sounds in reverse and in different speeds, using volume control to manipulate the intensity and the envelope of sounds and creating endless loops, which Schaeffer achieved through cutting the disc grooves so that the sounds would repeat, in a similar manner to a scratched disc. The success from *"Études de bruits"* and from other experimental works and broadcasts led the RTF to provide funds for the creation of the first audio studio in the world exclusively dedicated to the production of EM, the Groupe de Recherches Musicales (GRM), in 1951.

Pierre Schaeffer continued his work, trying new techniques, methodologies and tools (prominently the tape recorder), and working together with several composers. Schaeffer's influence on modern music is undeniable, as stated by one of his students that achieved international success, Jean Michel Jarre, in an interview for Radio France Internationale:

> "Back in the '40s, Schaeffer invented the sample, the locked groove - in other words, the loop -, the delay and the concept of reinjecting sounds. It was Schaeffer who experimented with distorting sounds, playing them backwards, speeding them up and slowing them down. He was the one who invented the entire way music is made these days." Dicale (2007)

1.1.2 Tape Music

Schaeffer was not alone on this EM revolution. The availability of the magnetic tape recorder after World War II made the creation of EM easier, which resulted in new EM studios in other European countries, in the United States and in Japan. One of these studios was the Studio for Electronic Music of the West German Radio in Cologne. While in GRM the aesthetic approaches to EM creation focused on using only recorded natural sounds as source material, the Cologne studio worked on creating music using only electronically synthesized tones. Despite these strong aesthetic differences, the mutually exclusiveness of the approaches from each studio disappeared due to an influx of composers to both studios which led to a more broadly stylistic and open-minded period of EM (Holmes, 2012).

In 1966 the magnetic tape studio was still the leading edge in EM technology. By this year, there were at least 560 documented institutional and private tape studios in the world. Only 40 per cent of these were sponsored by institutions and corporations, while the rest were privately owned (Holmes, 2012). During these years, the major centres such as Cologne, Paris and Columbia/Princeton were focusing their attention on producing works for tape. This included recording and manipulation of acoustic instruments as well as natural and synthesized sounds. The aesthetics, practices and techniques used by composers of tape music are still found in modern EM and laid the groundwork for its development. Analyzing these techniques and practices, Holmes (2012) proposes seven fundamental traits of EM, being number 5 the one most relevant to our motivation: “In EM, the sound itself becomes the material of composition”.

1.1.3 Digital Music Production

The developments in integrated circuit technology, specifically the invention of the microprocessor in the 70s, led to new technologies, as well as a price reduction, in EM equipment. New commercial and affordable digital samplers started to surface in the 80s, which allowed music makers to record and process sound without the need for a tape recorder. The first commercially available digital sampler was the Fairlight CMI (Holmes, 2012). Despite its high price (18000£ for the first version), it was able to reach commercial music, being used by artists such as Herbie Hancock and Stevie Wonder. Due to the popularity of this sampler, new samplers surfaced such as the Akai SP60 and the E-mu SP1200, which extended Fairlight CMI’s sampling capabilities, with an easier-to-use interface that allowed quick cutting and pasting of sound, at an affordable price. These samplers were able to reach a wider audience, democratizing music production and creating a big imprint in many genres, especially in Hip-Hop where these samplers and updated versions of them are still used. In this genre, musical excerpts, primarily from Funk and Soul music, such as drum breaks were processed, looped and rearranged for a rapper to sing on top. This is an example of derivative work, which we will detail in the next section.

The last stage on the democratization of music production started with the popularization of personal computers. The surfacing of DAW software for personal computers allowed the music maker to have the tools for composing, playing instruments and processing audio all in their personal systems. Along with this software, music makers may use controllers, such as MIDI keyboards or pad-based controllers, as well as plugins which include software audio synthesizers, software samplers and software effects. Currently, if an external audio interface with sufficient inputs or outputs is provided, a DAW can provide a large number of audio inputs or outputs, in the case of Ableton Live 10, 256 mono inputs and outputs, as well as unlimited audio and MIDI channels inside the software (Ableton, 2022). This availability of comprehensive software for music production, together with easy access provided by the internet to information on how to use them, has allowed people to start making EM without having the need for expensive hardware or even a professional recording studio. The majority of the EM done today is created or processed through a computer and very famous pieces have been created with minimal music making hardware, in places such as bedrooms or garages.

1.1.4 Large-Scale Sound Databases

The easy access to data sharing that the internet provided led to a series of social changes (Ritzer & Jurgenson, 2010). Consumers are shifting from a role of passive consumption to customers who produce. As stated in Tapscott (1996), “This is a dynamic world of customer innovation, where a new generation of producer-consumers considers the “right to hack” its birthright”. Lessig (2008) proposes the “Remix Culture”, a society which encourages derivative works i.e. works based on other copyrighted works. The Creative Commons license proposed by the Creative Commons organization, of which Lessig is one of the founders, enabled these derivative works by allowing content creators to authorize the free distribution, use and remixing of their work.

In the music ecosystem, the changes provided by the easy access to data sharing go from recording, to production, to distribution and to consumption. Websites like Bandcamp[2] and Soundcloud[3] replace the role of the music retailer and distributor by allowing independent artists or labels to release and dis-

[2]https://bandcamp.com
[3]https://soundcloud.com

tribute their music online, allowing the use of a Creative Commons license. In the case of music recording and production, a set of online digital tools are available to the producer-consumers. These range from online classes, to virtual instruments or effects, to recorded sounds to be loaded in a sampler. The latter might include premade audio loops, notes of recorded instruments or even one-shot hits of audio. Ratcliffe (2014) proposes a typology for sampled material in EDM which can be grouped into 4 categories: i) short, isolated fragments; ii) loops and phrases; iii) larger elements; iv) transformed material. DAWs normally come out of the box with a broad selection of sounds and, if the music maker desires to increase their library, there is a variety of databases that offer audio content that can be used for EMP.

A large-scale sound database is a collection of sounds that may or not be oriented towards EMP. One example of the latter are catalogs of ethnographic music recordings, including those of CREM[4] or the Alan Lomax Archive[5]. Although not specifically oriented for EMP, the sounds present in these databases can be sampled as larger elements, according to the typology proposed by Ratcliffe (2014). The databases which are more oriented towards music making also have different characteristics between themselves. It may be the genre for which the sounds are selected, types of sounds, the destination sampling software or even the way they are commercialised. Currently, we see a trend where companies which used to sell curated sample packs shift their content to subscription paid cloud databases[6].

Throughout this thesis, we will use several public collections of EM audio material. However, our main focus will be an audio collection which is deeply connected with the Remix Culture and with Creative Commons licensing, Freesound[7]: an online collaborative sound database where people with diverse interests share recorded sound samples under Creative Commons licenses (Font et al., 2013). This platform has originated and is still being developed in the Music Technology Group of Universitat Pompeu Fabra, where this research is carried out.

1.2 Motivation

The increasing amount of sounds and sample packs available to music makers comes at the cost of a harder content browsing experience. Commonly, music makers collect sample packs which come in various degrees of organization,

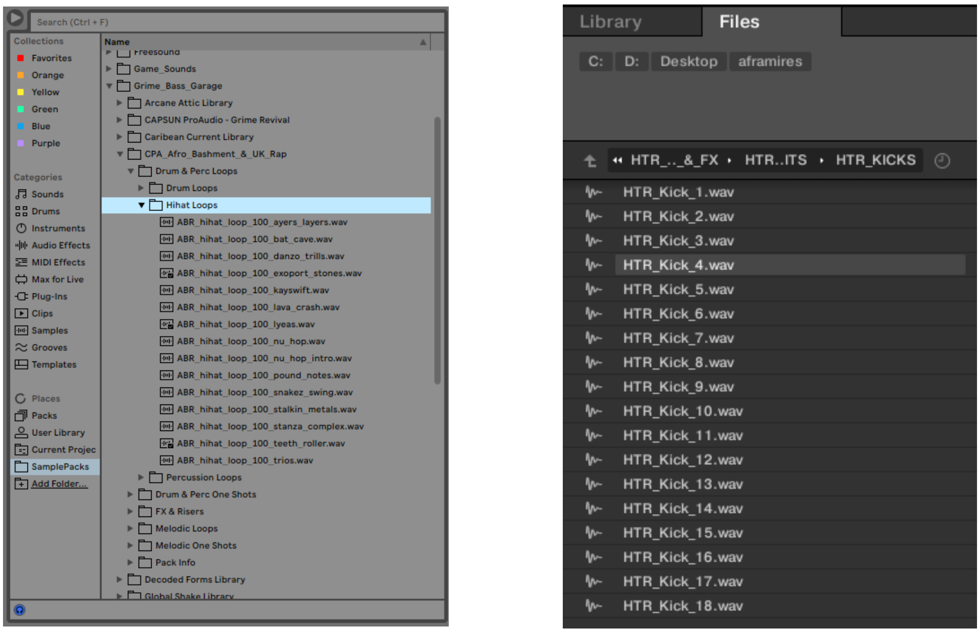
[4]https://archives.crem-cnrs.fr

[5]https://archive.culturalequity.org

[6]We can see examples of this in Native Instruments' sounds.com or Loopmasters' Loopcloud

[7]https://freesound.org

from a simple folder with several sounds inside to hierarchic directories with the sounds organized by characteristics such as their instrument, tempo or key. A personal collection can grow up to terabytes of audio files, and, if not well maintained, it can be chaotic to navigate through the different folders. We can already see from the sample browsing interfaces from two commonly used DAWs, presented in Figure 1.1, that the current hierarchical tree structure does not scale well with large collections of audio. Furthermore, each sample pack can have its own organization and naming strategies, which might be problematic for sound retrieval. Moreover, as sample packs are separated in different directories, very similar sounds from two different sample packs can be in different folders, leading to the music maker having to open and close several directories for finding the desired sound.

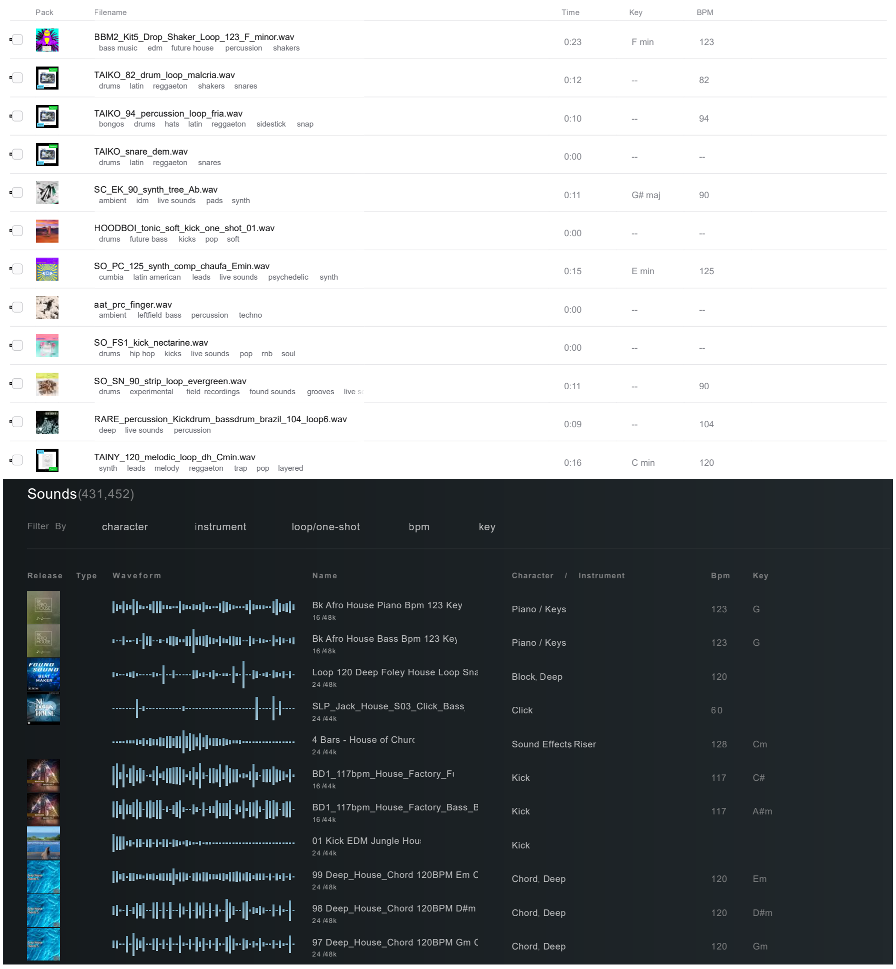


```
Freesound
Game_Sounds
Grime_Bass_Garage
  Arcane Attic Library
  CAPSUN ProAudio - Grime Revival
  Caribean Current Library
  CPA_Afro_Bashment_&_UK_Rap
    Drum & Perc Loops
      Drum Loops
      Hihat Loops
        ABR_hihat_loop_100_ayers_layers.wav
        ABR_hihat_loop_100_bat_cave.wav
        ABR_hihat_loop_100_danzo_trills.wav
        ABR_hihat_loop_100_exoport_stones.wav
        ABR_hihat_loop_100_kayswift.wav
        ABR_hihat_loop_100_lava_crash.wav
        ABR_hihat_loop_100_lyeas.wav
        ABR_hihat_loop_100_nu_hop.wav
        ABR_hihat_loop_100_nu_hop_intro.wav
        ABR_hihat_loop_100_pound_notes.wav
        ABR_hihat_loop_100_snakez_swing.wav
        ABR_hihat_loop_100_stalkin_metals.wav
        ABR_hihat_loop_100_stanza_complex.wav
        ABR_hihat_loop_100_teeth_roller.wav
        ABR_hihat_loop_100_trios.wav
      Percussion Loops
    Drum & Perc One Shots
    FX & Risers
    Melodic Loops
    Melodic One Shots
    Pack Info
  Decoded Forms Library
  Global Shake Library
```

```
HTR_Kick_1.wav
HTR_Kick_2.wav
HTR_Kick_3.wav
HTR_Kick_4.wav
HTR_Kick_5.wav
HTR_Kick_6.wav
HTR_Kick_7.wav
HTR_Kick_8.wav
HTR_Kick_9.wav
HTR_Kick_10.wav
HTR_Kick_11.wav
HTR_Kick_12.wav
HTR_Kick_13.wav
HTR_Kick_14.wav
HTR_Kick_15.wav
HTR_Kick_16.wav
HTR_Kick_17.wav
HTR_Kick_18.wav
```

Figure 1.1: Examples of how sample collections are presented on two DAWs: Ableton Live (left) and Native Instruments' Maschine (right)

When dealing with online collections, e.g. the websites of sample pack distributors or public sound collections, the problems are slightly different than the ones which occur in a personal library. Online collections normally contain larger amounts of sound, possibly tailored to different audiences and suitable to different genres. To handle the large diversity and quantity of sounds, these websites can resort to using a taxonomy to organize the sounds into different categories. Although this helps the navigation, each website has its own taxonomy which can create disorganization in the consumer's library. In Figures 1.2 and 1.3 we can see the user interface provided by three well-known online collections of sounds: Splice[8], Sounds.com[9] and Freesound[10].



	Pack	Filename	Time	Key	BPM
		BBM2_Kit5_Drop_Shaker_Loop_123_F_minor.wav bass music edm future house percussion shakers	0:23	F min	123
		TAIKO_82_drum_loop_malcria.wav drums latin reggaeton shakers snares	0:12	--	82
		TAIKO_94_percussion_loop_fria.wav bongos drums hats latin reggaeton sidestick snap	0:10	--	94
		TAIKO_snare_dem.wav drums latin reggaeton snares	0:00	--	--
		SC_EK_90_synth_tree_Ab.wav ambient idm live sounds pads synth	0:11	G# maj	90
		HOODBOI_tonic_soft_kick_one_shot_01.wav drums future bass kicks pop soft	0:00	--	--
		SO_PC_125_synth_comp_chaufa_Emin.wav cumbia latin american leads live sounds psychedelic synth	0:15	E min	125
		aat_prc_finger.wav ambient leftfield bass percussion techno	0:00	--	--
		SO_FS1_kick_nectarine.wav drums hip hop kicks live sounds pop rnb soul	0:00	--	--
		SO_SN_90_strip_loop_evergreen.wav drums experimental field recordings found sounds grooves live s	0:11	--	90
		RARE_percussion_Kickdrum_bassdrum_brazil_104_loop6.wav deep live sounds percussion	0:09	--	104
		TAINY_120_melodic_loop_dh_Cmin.wav synth leads melody reggaeton trap pop layered	0:16	C min	120

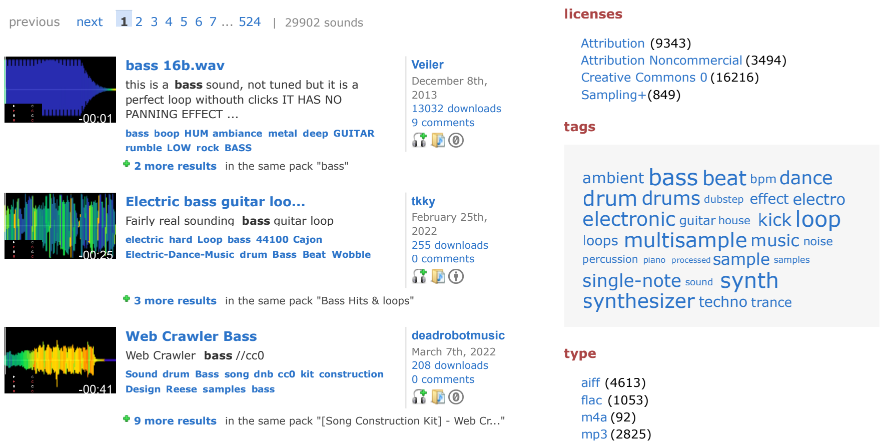
Sounds(431,452)

Release	Type	Waveform	Name	Character / Instrument	Bpm	Key
			Bk Afro House Piano Bpm 123 Key 16 /48k	Piano / Keys	123	G
			Bk Afro House Bass Bpm 123 Key 16 /48k	Piano / Keys	123	G
			Loop 120 Deep Foley House Loop Sna 24 /48k	Block, Deep	120	
			SLP_Jack_House_S03_Click_Bass_ 24 /44k	Click	60	
			4 Bars - House of Churc 24 /44k	Sound Effects Riser	128	Cm
			BD1_117bpm_House_Factory_Fu 16 /44k	Kick	117	C#
			BD1_117bpm_House_Factory_Bass_B 16 /44k	Kick	117	A#m
			01 Kick EDM Jungle Hou 24 /44k	Kick		
			99 Deep_House_Chord 120BPM Em C 24 /48k	Chord, Deep	120	Em
			98 Deep_House_Chord 120BPM D#m 24 /48k	Chord, Deep	120	D#m
			97 Deep_House_Chord 120BPM Gm C 24 /48k	Chord, Deep	120	Gm

Figure 1.2: User interfaces for Splice (top) and Sounds.com (bottom)

Online sample libraries targeted to EM creation such as Splice, Sounds.com or Loopmasters[11] have a fairly consistent organization and search interface. Curated sample packs are typically categorized into one or more genres, while individual samples can be found through the textual descriptions of their content. The instrument contained in the sample, timbral descriptors, genre, if it is a loop or a one-shot, tempo and key are tags commonly used to describe samples.

[8] https://splice.com
[9] https://sounds.com
[10] https://freesound.org
[11] https://loopmasters.com



previous next 1 2 3 4 5 6 7 ... 524 | 29902 sounds

-00:01 **bass 16b.wav**
this is a **bass** sound, not tuned but it is a perfect loop withouth clicks IT HAS NO PANNING EFFECT ...
bass boop HUM ambiance metal deep GUITAR rumble LOW rock BASS
Veiler
December 8th, 2013
13032 downloads
9 comments
2 more results in the same pack "bass"

-00:25 **Electric bass guitar loo...**
Fairly real sounding **bass** guitar loop
electric hard Loop bass 44100 Cajon Electric-Dance-Music drum Bass Beat Wobble
tkky
February 25th, 2022
255 downloads
0 comments
3 more results in the same pack "Bass Hits & loops"

-00:41 **Web Crawler Bass**
Web Crawler **bass** //cc0
Sound drum Bass song dnb cc0 kit construction Design Reese samples bass
deadrobotmusic
March 7th, 2022
208 downloads
0 comments
9 more results in the same pack "[Song Construction Kit] - Web Cr..."

licenses
Attribution (9343)
Attribution Noncommercial (3494)
Creative Commons 0 (16216)
Sampling+ (849)

tags
ambient bass beat bpm dance drum drums dubstep effect electro electronic guitar house kick loop loops multisample music noise percussion piano processed sample samples single-note sound synth synthesizer techno trance

type
aiff (4613)
flac (1053)
m4a (92)
mp3 (2825)

Figure 1.3: User interface for Freesound

Freesound is different from commercial sample databases due to its own nature. The first characteristic to highlight is the diversity of sounds available which goes from audio loops and instrument samples to be used in EMP, to foley sounds and field recordings. This diversity offers unconventional sounds to be used in EMP but also leads to difficult categorization and characterization and to a big diversity in the recording quality of sounds. The navigation and characterization of the sounds are based on unrestricted textual descriptions and tags of the sounds provided by users. This leads to a search based on noisy labels that different members use to characterize the same type of sound. Finally, the navigation is based on search queries which return a list of unorganized sounds.

Despite the differences between commercial sound collections, public libraries such as Freesound or the archive each music maker has, navigating and retrieving sounds can be cumbersome and lead to interruptions in the creative process. In Andersen & Knees (2016) the authors interview music makers to identify the difficulties they face when navigating their own personal sound collections. As expected, the organization of audio collections, indexing and efficient retrieval of sounds are central to their practice of creating music. Some of the issues they identify as problematic are presented in the following statements:

> "Because we usually have to browse really huge libraries [...] that most of the time are not really well organized."
>
> "If you have like a sample library with 500,000 different chords it can take a while to actually find one because there are so many possibilities."
>
> "Like, two hundred gigabytes of [samples]. I try to keep some kind of organization."

> “I easily get lost... I always have to scroll back and forth and it ruins the flow when you’re playing”

When asked about strategies to locate sounds ‘they are looking for’, music makers show a variety of strategies and emotions when dealing with DAWs:

> “You just click randomly and just scrolling, it takes for ever!”
>
> “Sometimes, when you don’t know what you are looking for, and you’re just going randomly through your samples, that might be helpful, but most of the time I have something in mind that I am looking for, and I am just going through all these sound files, and I am just waiting for the sound which I had in mind to suddenly appear. Or what comes the closest to what I had in mind. So I think that most of the time, I know what I am looking for, and then it is just a matter of time before I find it.”
>
> “Part of making music is about being lost a little bit and accidentally stumbling upon stuff that you didn’t think would work.”

From these fragments from the interviews conducted with music makers, we can clearly see that file retrieving is identified as the one of the things that mostly disrupts their workflow. In the same interviews, music makers ask for semantically meaningful retrieval systems, which should have features which enable surprise, opposition and control on the recommendation process:

> “So it would be really useful to for example have some kind of sorting system for drums, for example, where I could for example choose: ‘bass drum’, and here it is: ‘bass’ and ‘bright’, and I would like it to have maybe bass drum ‘round’ and ‘dry’, and you can choose both, the more I choose, of course, the less results I will have [...] So it is filtering it down, that is really helpful, if it works well of course.”
>
> “It would be even more useful to be able to search for a particular snare, but I can’t really imagine how, I need something short, low in pitch, dark or bright in tone and then it finds it...”
>
> “There are a lot of adjectives for sound, but for me, if you want a ‘bright’ sound for example it actually means a sound with a lot of treble, if you say you want a ‘warm’ sound, you put a round bass, well, round is another adjective.”

In the field of Music Information Retrieval (MIR), several studies were aimed at creating and evaluating interfaces for browsing and retrieving sounds. Techniques for presenting and arranging one shot-sounds, sorted by their timbral and sonic qualities, such as Freesound Explorer (Font & Bandiera, 2017) and Soundtorch (Heise et al., 2008), have been proposed. Significant research aimed at discovering the most relevant descriptors for audio artists, specifically the AudioCommons[12] project. One of the outcomes of this project was a set of high-level timbral descriptors which is derived from the most common adjectives used to describe sounds in Freesound. Most of the algorithms for calculating these descriptors were conceived by experts with deep knowledge on music and signal processing. If music makers can understand the features extracted from the audio, these can be directly used for navigation and characterizing their sample library. Another approach relies on using algorithms which identify similar content to group related sounds together (clustering) (Favory et al., 2020) or to present examples similar to a query (query-by-example) (Downie, 2003; Roma & Serra, 2015). Besides the descriptors, these techniques normally require a machine learning algorithm for handling the similarity computation. Machine learning algorithms can learn directly from the training data which are the most relevant features or correlation of features to describe the audio.

Deep learning methods are a subset of machine learning algorithms which are able to learn descriptors directly from the data, therefore removing the need for features designed by experts. This set of algorithms has shown impressive success in a variety of tasks, from the generation of images to music classification and auto-tagging. These technologies can be applied to audio samples collections, in order to assist musical creation through an improved sample browsing experience. However, deep learning algorithms typically require large amounts of data in order to be able to learn meaningful features and perform well. Large online collections with audio material for music making, such as Freesound, can be leveraged to obtain this data.

1.3 Scope and Objectives

The main goal of this dissertation is to develop novel data-driven systems which can enable new techniques for navigating large-scale collections and consequently assisting EMP. To this end, we leverage publicly available collections of musical audio aimed at music creation to train deep-learning models for two use-cases:

- Automatic Classification: systems aimed at automatically identifying characteristics which can help navigate large musical audio collections directly from the audio itself.

[12]https://audiocommons.org

- Generation: systems that are capable of creating new sounds similar to the ones in the collections, through the navigation of latent space. These systems enable generating random sounds, sounds similar to a query and interpolating between sounds.

This dissertation was conducted as part of the New Frontiers in Music Information Processing (MIP-Frontiers) project[13], which is a research project funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 765068. One of the main objectives of this project is to encourage collaboration between academia and industry so that the outcomes of the project can reach finished products and the end consumer in a faster manner. The host institution for this work is Universitat Pompeu Fabra, where Freesound is developed and maintained, while a secondment was conducted in Native Instruments GmbH (NI), which maintains a large collection of sounds in the factory library of their products, as well as in their expansions[14]. The use of Freesound and NI's data enables us to evaluate the proposed systems with two fairly different structured collections, Freesound which represents an unsorted library with different user methodologies for describing sounds and NI's data which is tagged by experts in a carefully thought hierarchy. Another mechanism for increasing the impact of research which is highlighted in the MIP-Frontiers project is making the data and code created during this project openly accessible and freely available. Towards this goal, we open-source all the code needed to replicate our experiments, as well as most of the data used throughout this work.

1.4 Dissertation Outline

In this section, we provide an overview of the following chapters, main experiments and contributions. The thesis is organized as follows:

- **Chapter 1** (this chapter) presents a brief history of EM and how repurposing audio material for composition has been one of its main elements. How audio material was used through time and the evolution of how audio files have been shared between music makers are introduced, as well as some of the common issues highlighted by creators when browsing their collections of audio files. This chapter also provides contextualization for the research conducted, its motivation and goals.

- In **Part I** we describe our contributions towards automatically characterizing instrumental audio samples:

[13]https://mip-frontiers.eu
[14]https://native-instruments.com/en/catalog/maschine/expansions

- **Chapter 2** introduces the topic of instrument classification, while **Chapter 3** presents a survey of previous techniques and datasets used for this task and some of its current shortcomings when applied to assisting EMP. Relevant deep learning algorithms for classification are introduced together with key audio signal processing concepts, which are explained so as to provide the reader with the necessary context to understand the following chapters.
- **Chapter 4** is based on Ramires & Serra (2019) and describes our work on training and evaluating a state-of-the-art model for classifying instrumental sounds when processed with commonly used audio effects in music production. In order to evaluate the robustness of the model, we use data augmentation with audio effects and evaluate how each effect influences classification accuracy.
- **Chapter 5**, based on Ramires et al. (2020c), presents the Freesound Loop Dataset (FSLD), a new large-scale dataset of music loops annotated by experts. The loops originate from Freesound, so the audio in the dataset may be redistributed. The annotations include instrument, tempo, meter, key and genre tags. The methodology used to assemble and annotate the data is described, as well as a report on the distribution of tags in the data and inter-annotator agreement. The online loop annotator tool that was developed for the annotation procedure is presented and, to illustrate the usefulness of FSLD, we describe short case studies on using FSLD to estimate tempo and key, generate music pieces, and evaluate a loop separation algorithm.
- **Chapter 6** is based on Drysdale et al. (2022) and details our work towards automatically labelling loops of key structural groups such as bass, percussive or melodic elements by the role they occupy in a piece of music through the task of Automatic Instrumentation Role Classification (AIRC). We experiment with several deep-learning architectures and propose a data augmentation method for improving multi-label representation to balance classes within the FSLD. To improve the classification accuracy of the architectures, we also evaluate different pooling operations. Additionally, we demonstrate how our proposed AIRC method is useful for analyzing the structure of Electronic Music (EM) compositions through loop activation transcription.

▪ In **Part II** we describe our contributions towards generating drum sounds, controlled by high-level parameters which music makers understand:

- **Chapter 7** provides a brief history of sound synthesis and the evolution of techniques for creating percussive sounds through time, while

  Chapter 8 presents an overview of neural networks used for synthesis, introduces commonly used architectures for this task, their advantages and disadvantages. A survey of deep learning models for synthesizing percussive sounds is also presented.

  - **Chapter 9**, based on Ramires et al. (2020b), presents a deep neural network-based methodology for synthesizing percussive sounds with control over high-level timbral characteristics of the sounds. This approach allows for intuitive control of a neural synthesizer, enabling the user to shape sounds without extensive knowledge of signal processing. We use a feed-forward convolutional neural network-based architecture, which is able to map input parameters to the corresponding waveform. Two datasets are presented to evaluate the approach in both a restrictive context and in one covering a broader spectrum of sounds. Finally, the quality of the output sound is evaluated using a subjective listening test.
  - **Chapter 10** is based on Chandna et al. (2021) and presents LoopNet, a feed-forward generative model for creating drum loops conditioned on intuitive parameters. This work extends Ramires et al. (2020b) to generate drum loops based on global features pertaining to timbre and harmony and time-varying features pertaining to the rhythm. We also evaluate the quality of the generated audio and propose intuitive controls and techniques for composers to map the ideas in their minds to an audio loop.
  - **Chapter 11**, based on Ramires et al. (2022), presents an analysis on evaluating how different deep learning based interfaces support creative control over drum generation by conducting a user study based on the Creativity Support Index (CSI). We train a Generative Adversarial Network (GAN) and experiment with both a supervised method that decodes semantic latent space directions and an unsupervised Closed-Form Factorization approach from computer vision literature to parameterize the generation.

- **Chapter 12** concludes this dissertation. We present a summary and a discussion of the work presented and show our progress towards the objectives we aimed to pursue. Future perspectives for research in the topic of assisting EMP with deep neural networks.

- Additionally, in **Appendix C** and **D** we present two codebases we developed: TIV.lib (Ramires et al., 2020a) and Freesound JUCE API (Ramires & Font, 2019). TIV.lib is an open-source python library for the content-based tonal description of musical audio signals based on the Tonal Interval Vectors (TIV) (Bernardes et al., 2016a). The Freesound JUCE API is a client library which interacts with the Freesound web API

(Akkermans et al., 2011) implemented in JUCE, a library commonly used for creating virtual instruments for DAWs.

Part I

Automatic Characterization of Music Loops and Instrument Samples for Electronic Music Production

CHAPTER 2

Introduction and Overview of Characterisation

In Part I (this Part) of this dissertation, we aim to use deep learning to classify instrumental sounds. Most commercial providers of sounds for music-making use expert annotation to classify and characterise the content they provide, which becomes costly. Collections such as Freesound and the personal libraries of music makers cannot afford this annotation service.

Our main goal is to create technologies which can analyse the audios present in unorganised collections of sound material and provide tags which can help music makers navigate these collections. Automatically classifying instrumental sounds in unstructured large audio databases provides an intuitive way of navigating them, and a better characterisation of the sounds contained. For databases where the annotation of the sounds is done manually, it can be a way to simplify the job of the annotator, by providing suggested annotations or, if the system is reliable enough, only presenting sounds with low classification confidence. The recent developments in deep learning towards automatically characterising unorganised data show that this set of algorithms is a promising avenue for us to reach our goal.

2.1 Automatic Audio Classification

The automatic classification of sounds is a heavily studied Signal Processing topic. It is applied to tasks we do in our daily life such as converting voice queries to text (speech recognition) and speaker identification. Automatic sound classification has been applied to several fields, some arbitrary examples of this are i) biology, where Kahl et al. (2021) present a system for identifying birds through recordings of their singing; in medicine, Laguarta et al. (2020) train an algorithm to discriminate a COVID cough from a normal one, and iii) sound event detection (Fonseca et al., 2019) which aims to classify the events that occur in an audio recording.

In the case of music, a strong effort has been put towards automatically extracting semantic characteristics which humans typically use to describe music pieces. Despite its musicology controversy, music genre classification has been a central task of MIR research with hundreds of publications written about the subject (Knees & Schedl, 2016). The aim of this task is to automatically assign a music genre or sub-genre, based on a taxonomy, to a music recording. It can help organise unstructured collections of music and, due to the largely available annotated collections of music belonging to streaming platforms, data-driven algorithms can be of great use.

A MIR task which is significantly related to genre classification is auto-tagging. Although both tasks aim to assign semantic labels to music pieces, auto-tagging labels are not specific to genre and sub-genre. The datasets used in auto-tagging typically dictate the tags a system predicts, with typical datasets used being the MagnaTagATune (Law et al., 2009), which contains tags proposed by amateurs in a game-like platform, and the Million Song Dataset (Bertin-Mahieux et al., 2011) where the tags have been obtained from scraping Last.fm[15]. Due to this diverse taxonomy, the auto-tagging classification is normally multi-label, where different tags such as "male vocal" and "hard rock" can occur together.

Another MIR classification task used to organise music libraries and automatically create playlists is mood and emotion recognition. This task employs a higher degree of subjectivity than the previous ones as the same music piece can create different emotions in each listener. The labels used for the classification can be categorical — a limited number of emotions such as happy, sad or angry — or dimensional — emotions are mapped to a continuous space with several dimensions of human emotions e.g. valence-arousal (Knees & Schedl, 2016).

2.2 Characterisation of Material for Music Production

In the previous section, we described MIR classification tasks which are useful for classifying finished music pieces but are not suitable for classifying music-making material. There are three MIR tasks which have seen a significant amount of studies that can help organise loops in unsorted collections: tempo, pitch, key and chord extraction.

[15]https://last.fm

Tempo estimation is a topic which has received significant attention in MIR (McKinney et al., 2007; Gouyon et al., 2006) and state-of-the-art algorithms are able to achieve high accuracy in identifying the tempo of music (Böck et al., 2015). Tempo is defined in Gouyon et al. (2006) as the *rate of musical beats in time*. It provides a useful feature for the retrieval of sounds for EMP when the music being created is based on a fixed tempo, which is the most typical scenario in EDM. The music maker will only work with a specific range of tempo values based on the tempo of the track being created so that no sound deformation occurs when time-stretching the audio to fit the desired tempo.

A common framework in tempo estimation approaches is proposed in Gouyon & Dixon (2005) which Böck et al. (2015) summarises in three stages and presents the techniques used in recently proposed algorithms. The first stage is the derivation of relevant features from raw audio such as a list of onset times or a frame-based feature vector. Previous examples have used the envelope of the audio signal (Scheirer, 1998), bandpass filters (Wu & Jang, 2014) and onset detection functions (Davies & Plumbley, 2007). The second step is the calculation of periodicities from the extracted features. For this task, Fast Fourier Transform (FFT) based methods like tempograms (Wu & Jang, 2014), autocorrelation (Dixon, 2001) or comb filters (Davies & Plumbley, 2007) have been used. Finally, a dominant period is extracted from the calculated periodicities through post-processing. The output of this final stage is the rate of beats of the audio material and can be expressed in Beats Per Minute (BPM). The techniques used for this last stage range from simply selecting the most dominant periodicity peak, to machine learning approaches and the use of music genre-specific knowledge to create constraints on the possible tempo measures (Davies & Plumbley, 2008). The state-of-the-art accuracy on several datasets is obtained by Böck et al. (2015), where the authors use neural networks to learn features to be used in the first stage of the tempo estimation, together with a resonant comb filter.

The evaluation of these algorithms is normally based on their accuracy in tempo estimation of annotated music datasets. Evaluation metrics normally used are *Accuracy1* and *Accuracy2*, where one or two tempo estimations are accepted as the correct BPM. In the Music Information Retrieval Evaluation eXchange (MIREX) a tempo extraction challenge exists since its conception, where tempo estimation algorithms are compared on their accuracy on selected datasets.

Pitch is the perceptual quality of a sound which is connected to the fundamental frequency of a sound. While the timbre of a sound is related to the harmonics of the fundamental frequency generated by an instrument, pitch allows judging sounds as higher or lower. Automatically identifying the pitch of a musical sound allows knowing the note played in a sound sample and enables posterior repitching to create a tuned melody. When applied to longer music sections, where there are several notes played in a sequence, this task is usually called pitch tracking (Salamon, 2013). Early works in predominant melody extraction focused on extracting the pitch of individual instrumental sounds (Anderson, 1997). After, works such as Salamon & Gómez (2012) and Mauch & Dixon (2014) have addressed the extraction of a monophonic lead melody from polyphonic recordings. Recently, the focus has shifted towards enabling the extraction of polyphonic melodies from polyphonic audio (Cuesta et al., 2020).

Another audio characterisation problem which has seen significant research in MIR is key estimation. In Gómez (2006a), key is defined as "a system of relationships between a series of pitches having a tonic, or central pitch class, as its most important element". The motivation of key estimation for EMP is of similar nature to tempo estimation. Music makers need tonal information to mix and layer sound files according to their tonal content (Faraldo et al., 2016).

Key estimation systems usually follow an essential architecture. We will only focus on the approaches which retrieve characteristics from the audio domain and not symbolic approaches. Faraldo Pérez (2018) details the most commonly used architectures for this task, which can be decomposed on two steps. First, the audio signal is converted to the frequency domain and mapped into vector representations which present the intensities of the twelve semi-tones of the pitch classes over time (Peeters, 2006), such as Chroma or Pitch Class Profile. This step is followed by a classification stage where these pitch classes are compared to reference material. In the case of template-based key estimation methods, these vector representations are then compared to existing tonality models, which can be key profiles (Peeters, 2006) or a geometrical space (Chew, 2000).

The evaluation of these systems, which also have a MIREX task, is also normally performed by quantifying their accuracy on key labelling an annotated dataset. In the evaluation of this MIREX task, incorrect key estimations which are considered tonally close are less penalised than the unrelated estimations.

Key estimation using contextual information from EDM has been explored in Faraldo Pérez (2018). In Faraldo et al. (2016), the authors present tonal characteristics of EDM which suggest that key estimation "should take into account style-specific particularities and be tailored to specific genres rather than aiming at all-purpose solutions". In his work, a new key profile is developed, trained on an EDM dataset, and some pre-processing related to the presented characteristics is proposed. The use of the context related to the music genre shows better performance than general approaches for an EDM dataset.

Another related task which deserves attention is chord estimation. A similar approach to key estimation is used for this task by firstly converting audio to a vector representation of the pitch classes and then a classifier is used to identify the chord. A detailed review of the methods used in literature for these steps is proposed in McVicar et al. (2014).

2.3 Automatic Instrument Classification

Instrument classification is a task which aims to identify the instrument that created a particular sound in an audio recording. To understand this task, we need to understand one musical concept which is essential to distinguishing instruments: the timbre. In the seminal instrument classification work by Herrera-Boyer et al. (2003), the authors employ the definition of timbre provided in American National Standards Institute. Committee on Bioacoustics et al. (1973): “the features that allow one to distinguish two sounds that are equal in pitch, loudness, and subjective duration”. Instrument classification has been targeted for different use cases such as instrument detection (understanding which instruments occur on a recording), instrument segmentation (identifying when certain instruments occur in an audio recording) or for assisting other MIR tasks such as source separation (Knees & Schedl, 2016; Schedl et al., 2014).

In Part I of this dissertation, we focus on classifying electronic music production sounds based on their timbral characteristics. We aim to apply and evaluate classification algorithms in the task of instrument and instrumentation role classification, towards a better characterisation of sounds in collections for electronic music making. Instrument classification has mostly targeted the classification of acoustic or electrical instrumental sounds in isolation or in music recordings. Using deep learning and taking advantage of the ease of access to digitally generated sounds from virtual synthesizers and online collections, we want to use instrument classification for characterising sounds used in EMP. EM sounds can have very different characteristics from the acoustic and electric ones which were studied in classic MIR due to the possibilities offered by synthesizers, heavy use of sound effects and the freedom of exploration in sound design favoured in EM.

In this chapter (Chapter 2), we presented existing techniques for audio classification in MIR and motivated as to why specific algorithms suited for EM are necessary. In Chapter 3 we will present an overview of the methodologies for instrument classification used on the past and present the deep learning context required to understand the approaches we use in the subsequent chapters. In Chapter 4, we present our work towards classifying the instruments which created the one-shot sounds from a big collection of synthesised notes. Data augmentation techniques based on audio effects and their performance when classifying sounds processed by audio effects are also presented. Chapter 5 presents our effort in creating the Freesound Loop Dataset (FSLD), a Creative Commons collection of audio loops meant to be used in EMP. The annotation tool developed for collecting and describing the loops is presented, as well as several benchmarks for MIR tasks. Finally, Chapter 6 presents our work in Automatic Instrumentation Role Classification (AIRC) in loops. Instead of automatically identifying instruments, we aim to find if a loop contains chords, percussion, melody, effects and/or bass. We also extend this work in order to automatically infer the structure of EM pieces.

CHAPTER 3

Literature Review on Instrument Classification

In this chapter, we make a review of the key ideas proposed for instrument classification through time. In order to provide a better understanding of the machine learning methodologies that have been used in previous work and the deep learning algorithms we use throughout our thesis, we will present an introduction to how these data-driven algorithms work, their main weaknesses and advantages.

We will start by presenting an overview of machine learning methods. Then, we will focus on deep learning algorithms for classification and finalize with an in-depth look at instrument classification approaches.

3.1 Machine Learning and Automatic Classification

Machine learning is the discipline which studies how to build and understand systems that can automatically improve and learn through experience (Jordan & Mitchell, 2015). Machine learning approaches enable us to train a system by providing it data examples with the desired input-output, which for several tasks is easier than designing the system manually to do the desired task. Therefore, machine learning can enable the development of complex algorithms without needing expert knowledge of the data or task at hand.

Machine learning can be divided into three main paradigms: supervised, unsupervised and reinforcement learning. As the main focus of Part I of the thesis is supervised learning, we will start by providing an overview of the two other concepts and then focus on supervised learning. Reinforcement learning uses agents which perform actions, which can be correct or not, to learn how to interact with an environment to achieve a goal (Sutton & Barto, 2018). One easy-to-understand application of this concept is the Artificial Inteligence (AI) for games which learn to be good at a game by playing against other AIs. Different AIs can have different exploration and exploitation settings, and a set of actions can be deemed correct if an AI can win a game. The AI which won more games can be selected as the best one and used to play against real players. Instead of working with an expert player at the game to develop a rule-based algorithm that can win a game, we can train one of these AIs at a lower cost and possibly better results. Reinforcement learning is used in several fields such as autonomous driving, sequence prediction and gaming. If the reader would like to know more about this concept, we recommend reading the work of Sutton & Barto (2018), which provides a good introduction to this topic.

Unsupervised machine learning algorithms will be studied in more detail in Part II. However, we will now provide an overview to more easily differentiate it from supervised learning. Unsupervised learning generally involves learning patterns from unlabeled data. One of the most well-known tasks in this topic is dimensionality reduction. By training dimensionality reduction algorithms with high-dimensionality data such as images (which are normally represented by a huge amount of pixels) or sound (which can have more than 44.1 thousand samples per second), we can ideally obtain a compact representation of the examples in, for instance, 2- or 3-dimensional visualizations. Well-known algorithms for dimensionality reduction include Principal Component Analysis (PCA) (Wold et al., 1987) and Autoencoder (AE) (Kingma & Welling, 2014) that we will review in Part II. Another typical problem in unsupervised learning is automatic clustering — partitioning unlabeled data into smaller groups with similar characteristics. Examples of clustering algorithms are k-Means and Hierarchical Clustering. An exhaustive list of clustering algorithms is available at Xu & Tian (2015). The main issue with unsupervised learning is that the amount of data necessary for these algorithms to learn high-quality representations is significantly bigger than when using labeled data (Jordan & Mitchell, 2015).

The most widely used machine learning methods and the ones that we will be exploring in Part I of this thesis are supervised machine learning algorithms. As the name indicates, these algorithms learn from labeled training data. This ground truth data takes the form of input-output pairs, which we will represent by (x,y). The objective of this algorithm is to learn a function that maps the inputs x to an output prediction y. Ideally, the model can learn from the training data and be able to predict an output correctly y^* for an input x^*, which was not seen during training. The input x can be a simple vector or take a more complex representation such as text, an image, a sequences or a sound. Likewise, the output y can go from a simple binary True or False (e.g. in the case of spam detection) to a vector of binary labels or even images and sound. Supervised learning is typically divided into two main problems according to their output: regression when predicting continuous values and classification when predicting a discrete class label.

In classification tasks, the aim is to create a program which can assign one or more of k categories to an input x. When using machine learning, a trainable algorithm is usually given the objective to produce a function $f : x^* \in \mathbb{R}^n \rightarrow 1, ..., k$ (Goodfellow et al., 2016). This classification algorithm can be represented by $y^* = f(x^*)$, where the output y^* is a combination of the k possible labels. If the algorithm is supposed to only provide a single label to a sound, the task is usually named single label classification. Otherwise, if more than one label can be assigned, the task is called multi label classification. In some cases, the classification algorithm outputs the probability of the input x^* belonging to any of the classes. To convert the probability to a binary label, a binary decision threshold can be used.

Typically, the pipeline of classification algorithms can be divided into two parts: the front- and back-end. When using this terminology, the front-end is responsible for directly interacting with the input and mapping it through a feature extraction part, i.e. extracting relevant patterns or characteristics from the data. The output of the front-end part can be a high-dimensional embedding space of handcrafted features[16] carefully selected by a researcher or a latent space learned by the machine learning algorithm itself. The back-end is the part responsible for mapping the representation given by the front end to the output label. Traditionally, classification with machine learning used feature extraction front-ends. As an example in the audio domain, if we would like to classify the instrument of a sound, we would start by selecting handcrafted audio features pertaining to timbre such as the spectral flux, spectral complexity or the Mel-Frequency Cepstral Coefficients (MFCC) from Essentia (Bogdanov et al., 2013) and extract these features from our audio dataset. Then, we could use these as the input to a classifier such as k-nearest neighbors, Support-Vector Machines (SVMs) or decision trees[17]. The idea behind this rationale is that by reducing the input from the digital audio which can have a dimensionality of $SampleRate * Duration$, these algorithms are easier and faster to train and we keep a pertinent audio representation for the task. These classifiers typically use an objective function with trainable variables (also called weights) whose values are set by iterating through the dataset. By going through each example (x, y) on the training dataset, the classifiers adjust the value of the weights to find the best fit for $y = f(x)$ in the full dataset.

Machine learning algorithms come with their own set of issues. The main ones are related to the data they are trained with. If this data is insufficient,

[16]By handcrafted features, we mean characteristics designed by experts, which are derived from the input without the use of machine learning.

[17]An explanation of how these algorithms and a complete overview of classification algorithms can be found in Mitchell (1997).

lacks variety or does not represent the problem at hand, the algorithm is due to overfitting — not being able to generalize well to data never seen before. Furthermore, some of these algorithms, especially ones with lots of trainable weights, lack interpretability, as the mapping $f(x)$ between input and output can be very abstract. Moreover, if the data is not representative of what it samples, the algorithms can have biases towards lower represented classes. For very complex problems, extracting features which are tailored to all the possible data variations can be close to impossible using handcrafted features. For example, in speech recognition, it would be very complicated to create handcrafted features which are invariant to the speaker's accent (Goodfellow et al., 2016).

In the following section, we will look at a new set of machine learning algorithms which aim to bypass this feature designing problem: deep learning algorithms.

3.2 Deep Learning Architectures for Classification

With the increase of computational power in the past years, a new set of algorithms within machine learning have been proposed. These are called deep learning algorithms and they are able to learn directly from the high-dimensional data, therefore skipping the handcrafted feature selection step. Although deep learning algorithms have existed since the 1940's (Goodfellow et al., 2016), their success and popularity is relatively new. This is due to several factors but mostly to the increase in the amount of available training data and the increase in computational power and software libraries for parallel processing of data.

Neural networks are the backbone of deep learning algorithms. While some neural networks, such as the feed-forward neural network, have been used as back-ends in the traditional machine learning paradigm, they excel when used within the deep learning paradigm. The name of these networks comes from their inspiration from the architecture of the brain, where neurons are connected to each other to create a large network. To understand neural networks, we will start by understanding their most basic component — the neuron — and understand how they connect to create networks. We will then look at commonly used neural network architectures and understand some of their main issues.

The artificial neuron consists of an affine transformation which is followed by a non-linearity (Bengio, 2009). They have several inputs and only one output which can be broadcast to other neurons. The affine transformation consists of a weighted sum of all its inputs and a bias term. Both the weights of the sum and the bias are trainable parameters. The non-linearity step is achieved by

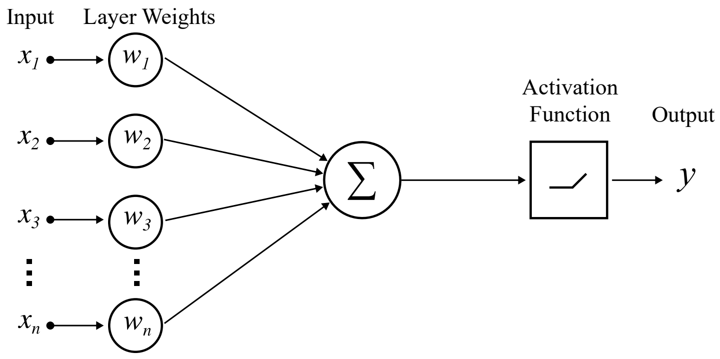


Figure 3.1: Example of an artificial neuron used for creating layers in neural networks.

an activation function which should be differentiable so that a gradient-based training can be used and computationally efficient for fast training. Commonly used activation functions are softmax (3.1), Rectified Linear Units (ReLU) (3.2)(Nair & Hinton, 2010) or a sigmoid function (3.3). A visual representation of an artificial neuron is provided in Figure 3.1.

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

$$f(x) = max(o, x) \tag{3.2}$$

$$f(x) = \frac{1}{1 + \exp -x} \tag{3.3}$$

In feed-forward networks, neurons are organized into multiple layers, where each neuron receives its input from the neurons of the previous layer and sends its output to the ones in the next layer, as can be seen in Figure 3.2. The first layer is typically called the input layer and is responsible for receiving external data. Likewise, the output layer is the last one and is responsible for outputting the classification result. The layers in between are termed hidden layers and these are responsible for learning the representations needed to go from input to a label. Typically, the closer a layer is to the output layer, the higher the level of the representation calculated (Bengio, 2009).

As we have seen before, the goal of a classification algorithm is to approximate a function $y = f(x)$, which maps an input x to a category y for all the examples in the training data. In a feed-forward network, if the weights are represented by $\boldsymbol{\theta}$, the approximated mapping function can be represented by $y^* = f^*(x, \boldsymbol{\theta})$. The goal of the network is to learn the best $\boldsymbol{\theta}$ so that $f^*(x, \boldsymbol{\theta}) \approx f(x)$. To achieve

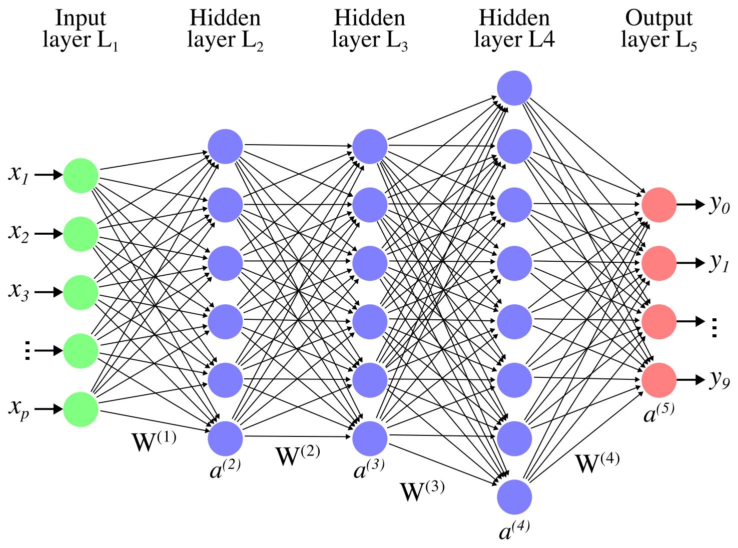


Figure 3.2: Example of a neural network layer.

this goal, the training procedure used in most deep learning algorithms is backpropagation based on Stochastic Gradient Descent (SGD) optimization. For a batch of examples from the training dataset, an error is calculated between the network's output and the ground truth labels using a loss function. This loss function is normally designed to fit the task at hand, where the researcher chooses which kind of errors are more or less significant in the application scenario. Commonly used loss functions for classification are the mean-squared error (3.4) or the binary cross entropy loss (3.5). When the loss is calculated for a batch of examples, the next step is to adjust the weights based on this error. This is where the SGD acts by identifying which weights were more responsible for the error and by modifying them by a proportional amount. An important parameter to introduce here is the learning rate, a tuning parameter which defines how much the weights should be modified. Setting a high value for this parameter improves convergence speed but leads to lower final accuracy. On the other hand, if this value is set too small, it might take too long to train and can make the optimization procedure get stuck in local minima. A common technique is to use variable learning rates based on training time or classification accuracy.

$$f(y,\hat{y}) = (\hat{y} - y)^2 \tag{3.4}$$

$$f(y,\hat{y}) = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y})) \tag{3.5}$$

Another common technique applied to training neural networks is regularization: modifications to the algorithm in order to reduce its generalization error but not its training error (Goodfellow et al., 2016). These techniques are fairly different from each other, as this is a very generic problem. However, some commonly used examples are:

- Data augmentation: artificially creating variations of the training data, e.g. rotating the images in a dataset;
- Parameter norm penalties: constraining the weights so that they take values within a range;
- Dropout: randomly disabling neurons during training so that the network learns to be more robust;
- Early stopping: stopping the training before the accuracy is maximum in the training set
- Noise robustness: making the model better suitable to deal with noisy data by injecting it at the label level or in the weights.

Finally, an important step for neural network training is the initialization of the weights. If the initial values are too large, the network will not learn well and will have exploding gradients. If they are too small, the network will not learn at all. Two algorithms are typically used for the initialization: Xavier (Glorot & Bengio, 2010) and He (He et al., 2015).

A detailed review of neural network training, loss functions, initialization, regularization and learning rate schedule is provided in Goodfellow et al. (2016).

A special type of neural networks which are particularly suited to sequential data and which have seen increasing interest in classification tasks are Convolutional Neural Network (CNN). Instead of neurons, these networks use a function inspired by the convolution mathematical operation to extract information from the data. Instead of learning the weights of neurons, these networks learn filters (also known as kernels) which slide through the data and identify relevant features. After performing this convolution-like operation with a kernel through an input, we obtain an intermediate representation often called feature maps. In Figure 3.3, we show an example of the architecture of a CNN with feed-forward output layers.

Typically, a convolutional layer has three components: the convolution stage with several filters, followed by a nonlinear activation function and a pooling stage. Pooling is a special type of operation which is heavily used in CNNs for reducing the size of feature maps and, consequently, memory usage, number of parameters and computational complexity. To this end, they take parts of the feature maps and reduce them to a smaller representation. There are

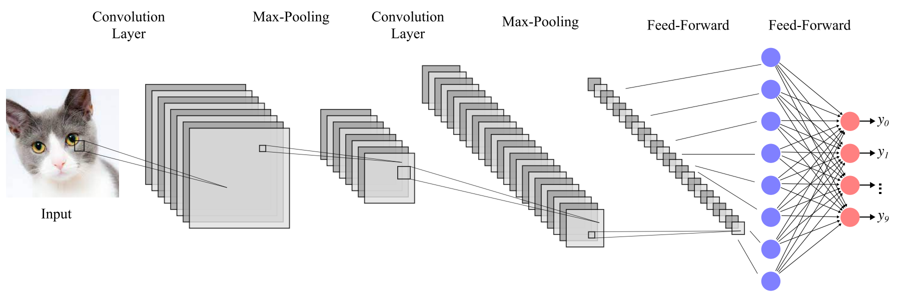


Figure 3.3: Convolutional neural network architecture with feed-forward output layer.

two heavily used pooling operations which are used: max and average pooling. Max and average pooling take a window of fixed size ($2x2$ is commonly used) and slide it through the feature map. Their output is either the maximum or average value in that windowed part of the feature map. Global pooling is a similar technique which uses the full feature map instead of a window. The use of pooling layers helps making the learned representations invariant to translations in the input (Goodfellow et al., 2016).

When training classification neural networks, it's common to split the full data we have into 3 groups. The training set is used for the network to learn the representations and classification weights. The validation set is a part of the data that the algorithm does not see during training. The classification accuracy on the validation set provides us with information on how the model performs on unseen data and allows us to tune the parameters of the model to achieve a higher generalization. Finally, the test set is assembled from the data which is not present in the training and validation set and enables us to obtain an approximate generalization error and overall accuracy of the trained model.

The biggest issues with deep learning models relate to their use of big amounts of data and processing power, as well as the lack of interpretability of their results. As we know, deep learning models benefit from the amount of data available for training. Access to enormous amounts of data and, consequently, computational capability is normally reserved for big companies. Although they are responsible for some of the most fascinating applications which use deep learning, the lack of resources in smaller companies and universities makes this field very unbalanced. Interpretability of the results provided by deep learning methods is also a very significant problem. Although some recent studies have been delving into increasing the interpretability of deep learning models, these algorithms are hard to decipher due to the abstractions that form their basis. Finally, deep learning models tend to reproduce the same biases present in their data. Although this is normally an issue that derives from the biases in the real world, we want deep learning algorithms to be fair and treat everyone the same. For an in depth survey on algorithmic biases and fairness concerns in deep learning, we refer the reader to Mehrabi et al. (2021).

3.3 Instrument Classification

Automatic instrument classification can be split into two related tasks with a similar goal. The first is the identification of instruments in single instrument recordings (which can be isolated or overlapping notes), while the second is recognising the predominant instrument in a mixture of sounds.

Instrument classification approaches followed a similar course as general classification methodologies. Early algorithms used two modules for this classification, one for extracting and selecting handcrafted features (e.g. MFCC, spectral centroid, roll-off, and flux) and another for classification (e.g. k-nearest neighbors, SVMs or hidden Markov models). For example, Essid et al. (2006) experiment with more than 150 handcrafted features in which a selection is given as input to Gaussian mixture modeling algorithm and SVMs. Benetos et al. (2006) perform instrument classification by combining a set of handcrafted features, a feature selection algorithm and non-negative matrix factorization. In Fuhrmann & Herrera (2010), SVMs with a set of timbral-related handcrafted features are used to quantify music similarity. A thorough overview of algorithms for instrument classification using traditional machine learning techniques is presented in Herrera-Boyer et al. (2003) and Fuhrmann (2012).

Recent work has shown the effectiveness of using CNNs for instrument classification (Pons et al., 2017b; Han et al., 2017; Li et al., 2015; Park & Lee, 2015). Commonly used input representations of audio for instrument classification are the waveform and the spectrogram of the audio. While the raw audio can provide all the information possible to the network, for long inputs, the network has to be very big. On the other hand, when dealing with spectrograms, the input size can be smaller, and the networks are, therefore, more compact. When raw audio or spectrograms are given, CNNs are able to learn and identify local spectro-temporal patterns relevant to the task to which they are applied. When utilized for MIR tasks, CNNs have outperformed the previous state-of-the-art approaches for various tasks (Nam et al., 2019; Han et al., 2017). For automatic instrument classification, the state-of-the-art approaches use CNNs trained on different representations of the input, such as raw audio (Li et al., 2015), spectrograms together with multiresolution recurrence plots (Park & Lee, 2015) and log Mel-frequency spectrograms (Han et al., 2017; Pons et al., 2017b). In Pons et al. (2017b), CNNs were tailored towards learning timbre representations in log Mel-frequency spectrograms by using vertical filters instead of the commonly used square filters. This provides a pitch invariance on the features learned by the front end, which is ideal for timbre-related features. For instrument classification, this approach displays a close to the state-of-the-art (Han et al., 2017) accuracy on the Instrument Recognition in Musical Audio Signals (IRMAS) dataset (Bosch et al., 2012) while reducing the number of trainable parameters by approximately 23 times on the single-layer proposed model.

The evaluation in instrument classification is a fairly simple task. The typical accuracy metrics can be used to measure the correctness of a model's prediction: f-measure, precision, recall and overall accuracy in case the model output is binary. In the case of a mode outputs a probability, Area Under the Receiver Operating Characteristic Curve (ROC-AUC) or Area Under the Precision-Recall Curve (PR-AUC) can be used. The Datasets used for the evaluation and training of these algorithms include Real World Computing (RWC) Music Database (Goto et al., 2002) or the University of Iowa Musical Instrument Samples[18]. While these datasets are small (RWC has 50 instruments), they proved to be good for classification using handcrafted features. New datasets such as the IRMAS Bosch et al. (2012) for predominant instrument classification and GoodSounds (Romani Picas et al., 2015) with single instrument recordings have been created and provided sufficient data for deep learning approaches to be able to surpass more traditional machine learning approaches. However, most of these datasets use either polyphonic music recordings or recorded notes of classical music instruments. In electronic music, a lot of the sounds present in music makers' collections are generated by synthesizers which provide different timbres than the ones in classical instruments. A new high-quality dataset of one-shot instrumental notes was presented in Engel et al. (2017), largely surpassing the size of the previous datasets, containing 305979 musical notes with unique pitch, timbre and envelope generated from digital instruments.

In Chapter 4, we will be looking at using this dataset to train a model based on the Pons et al. (2017b) CNN and augmenting it with audio effects to verify how robust it is to the processed data. In Chapter 5, we will present a new dataset of loops from Freesound, which can be used to properly evaluate how classification algorithms perform on real music-making data. Finally, in Chapter 6, we will use the loop dataset from Chapter 5 to train a classifier of instrumentation roles. This enables us to organize a loop collection by their instrumentation roles, and we show that it can be extended to automatically infer the structure of music compositions.

[18]https://theremin.music.uiowa.edu/MIS.html

CHAPTER 4

Classification of One-Shot Sounds

Automatic classification of one-shot instrumental sounds allows automatically categorizing the sounds contained in sound collections for electronic music making, allowing easier navigation and better characterization. Automatic instrument classification has mostly targeted the classification of unprocessed isolated instrumental sounds or detecting predominant instruments in mixed music tracks. For this classification to be useful in audio databases for Electronic Music Production (EMP), it has to be robust to the audio effects applied to unprocessed sounds.

In this chapter, we evaluate how a state-of-the-art model trained with a large dataset of one-shot instrumental sounds performs when classifying instruments processed with audio effects. In order to evaluate the robustness of the model, we use data augmentation with audio effects and evaluate how each effect influences the classification accuracy.

This chapter is based on **Ramires, A.**, & Serra, X. (2019). Data augmentation for instrument classification robust to audio effects. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

4.1 Introduction

The automatic classification of one-shot instrumental sounds remains an open research topic for Music Information Retrieval (MIR). While the research on this field has been mostly performed on clean and unprocessed sounds, the sounds provided by EMP databases may also contain "production-ready" sounds, with audio effects applied on them. Therefore, in order for this automatic classification to be reliable for EMP sample databases, it has to be robust to the types of audio effects applied to these instruments. In our study, we evaluate the robustness of a state-of-the-art automatic classification method for sounds with audio effects, and analyze how data augmentation can be used to improve classification accuracy.

Recent work has shown the effectiveness of using Convolutional Neural Network (CNN) for instrument classification (Pons et al., 2017b; Han et al., 2017; Li et al., 2015; Park & Lee, 2015). CNNs can be seen as trainable feature extractors, where kernels (or filters) with trainable parameters are convolved over an input, being able to capture local spatial and temporal characteristics. This architecture has been applied with great success to the detection, segmentation and recognition of objects and regions in images (LeCun et al., 2015). In the audio domain, when raw audio or spectograms are given, CNNs are able to learn and identify local spectro-temporal patterns relevant to the task to which they are applied. CNNs have outperformed previous state-of-the-art approaches for various MIR tasks (Nam et al., 2019; Han et al., 2017), specially instrument classification (Li et al., 2015; Park & Lee, 2015; Pons et al., 2017b).

Within the context of NSynth (Engel et al., 2017), a new high-quality dataset of one shot instrumental notes was presented, largely surpassing the size of the previous datasets, containing 305979 musical notes with unique pitch, timbre and envelope. The sounds were collected from 1006 instruments from commercial sample libraries and are annotated based on their source (acoustic, electronic or synthetic), instrument family and sonic qualities. The instrument families used in the annotation are bass, brass, flute, guitar, keyboard, mallet, organ, reed, string, synth lead and vocal. The dataset is available online[19] and provides a good basis for training and evaluating one shot instrumental sound classifiers. This dataset is already split in training, validation and test sets, where the instruments present in the training set do not overlap with the ones present in validation and test sets. However, to the best of our knowledge, no methods for instrument classification have so far been evaluated on this dataset.

In order to increase the generalization of a model further than the data provided to it, one possible approach is to use data augmentation. This approach can be described as applying deformations to a collection of training samples, in a way that the correct labels can still be deduced (McFee et al., 2015a). In computer vision, transforming images by cropping, rotation, reflection or scaling are commonly used techniques for data augmentation. In the audio domain, an intuitive and practical transformation is applying audio effects to the original training audio files. Transformations such as time-stretching, pitch-shifting, dynamic range compression and adding background noise have been applied with success to environmental sound classification, for overcoming the data

[19]https://magenta.tensorflow.org/datasets/nsynth

scarcity problems (Salamon & Bello, 2017). In Ko et al. (2017), artificial reverberation was applied to speech recordings, so as to create a speech recognition system robust to reverberant speech. For instrument recognition, the same set of effects used in Salamon & Bello (2017) was applied with success in McFee et al. (2015a). We believe that the use of audio effects typically used in EMP such as echo, reverb, chorus, saturation, heavy distortion or flanger can lead to a useful augmentation, as well as to an increase in robustness in instrument classification scenarios where the instrument recordings have these effects applied.

The rest of the chapter is structured as follows: Section 4.2 presents the methodology we used to classify the sounds in the dataset and the data augmentation procedure. In Section 4.3 we detail the evaluation of the different models, while Section 4.4 presents the results we achieved and a discussion of them. Finally, Section 4.5 concludes this chapter by presenting its contributions and proposes avenues for future work.

4.2 Methodology

In our study we conduct two experiments. First, we try to understand how augmenting a dataset with specific effects can improve instrument classification and secondly, we see if this augmentation can improve the robustness of a model to the selected effect.

To investigate this, we process the training, validation and test sets of the NSynth (Engel et al., 2017) dataset with audio effects. A state-of-the-art deep learning architecture for instrument classification (Pons et al., 2017b) is then trained with the original training set, and appended with each of the augmented datasets for each effect. We use the model trained with the original training set as a baseline and compare how the models trained with augmented versions perform on the original test and on the augmented versions of it for each effect. The code for the experiments and evaluation is available in a public GitHub repository[20].

4.2.1 Data Augmentation and Pre-Processing

The audio effects for the augmentation were applied directly to the audio files present in the training, validation splits of the NSynth dataset (Engel et al., 2017). For the augmentation procedure, we used a pitch-shifting effect present in the LibROSA[21] library and audio effects in the form of Virtual

[20]https://github.com/aframires/instrument-classifier
[21]https://librosa.github.io/librosa

Studio Technology (VST) audio plugins. For the augmentation which used audio plugins, the effects were applied directly to the audio signals using the Mrs. Watson[22] command-line audio plugin host. This command line tool was designed for automating audio processing tasks and allows the loading of an input sound file, processing it using a VST audio effect and saving the processed sound. In order to maintain transparency and reproducibility of this study only VST plugins which are freely distributed online were selected. The parameters used in the augmentation procedure were the ones set in the factory default preset for each audio plugin, except for those which the default preset did not alter the sound significantly.

The audio effects used were the following:

- **Heavy distortion:** A Bitcrusher audio effect which produces distortion through the reduction of the sampling rate and the bit depth of the input sound was used in the training set. The VST plugin used for augmenting the training set was the TAL-Bitcrusher[23]. For the test and validation set, we used Camel Audio's CamelCrusher[24] plugin which provides distortion using tube overdrive emulation combined with a compressor.

- **Saturation:** For this effect, tube saturation and amplifier simulation plugins were used. The audio effect creates harmonics in the signal, replicating the saturation effect from a valve- or vacuum-tube amplifier (Zölzer, 2011). For this augmentation we focused on a subtle saturation which did not create noticeable distortion. The plugin used in the training set was the TAL-Tube[23], while for the validation and test set Shattered Glass Audio's Ace[25] replica of a 1950s all tube amplifier was used.

- **Reverb:** To create a reverberation effect, the TAL-Reverb-4 plugin[26] was used in the test set. This effect replicates the artificial reverb obtained in a plate reverb unit. For the validation and test set we used Oril-River[27] algorithmic reverb, which models the reverb provided by room acoustics. The default preset for this plugin mimics the reverb present in a small room.

- **Echo:** A delay effect with long decay and with a big delay time (more than 50ms) (Zölzer, 2011) was used to create an echo effect. We used the

[22]https://github.com/teragonaudio/MrsWatson
[23]https://tal-software.com/products/tal-effects
[24]https://audiopluginsforfree.com/camelcrusher
[25]https://shatteredglassaudio.com/product/103
[26]https://tal-software.com/products/tal-reverb-4
[27]https://kvraudio.com/product/orilriver-by-denis-tihanov

TAL-Dub-2[28] VST plugin in the training set and soundhack's ++delay[29] in the validation and test set. For this last plugin, we adapted the factory default preset, changing the delay time to 181.7 ms and the feedback parameter to 50%, so that the echo effect was more noticeable.

- **Flanger:** For this delay effect, the input audio is summed with a delayed version of it, creating a comb filter effect. The time of the delay is short (less than 15 ms) and is varied with a low frequency oscillator (Zölzer, 2011; Reiss & McPherson, 2014). Flanger effects can also have a feedback parameter, where the output of the delay line is routed back to its input. For the training set, the VST plugin used was the TAL-Flanger[23], while for the test and validation sets we used Blue Cat's Flanger[30], which mimics a vintage flanger effect.

- **Chorus:** The chorus effect simulates the timing and pitch variations present when several individual sounds with similar pitch and timbre play in unison (Reiss & McPherson, 2014). The implementation of this effect is similar to the flanger. The chorus uses longer delay times (around 30 ms), a larger number of voices (more than one) and normally does not contain the feedback parameter (Zölzer, 2011; Reiss & McPherson, 2014). The VST effect used in the training set was the TAL-Chorus-LX[31] which tries to emulate the chorus module present in the Juno 60 synthesizer. For the test and validation sets, we used Blue Cat's Chorus[32], which replicates a single voice vintage chorus effect.

- **Pitch shifting:** For this effect, the LibROSA Python package for musical and audio analysis was used. This library contains a function which pitch shifts the input audio. As the dataset used contains recordings of the instruments for every note in the chromatic scale in successive octaves, our approach focused on pitch-shifting in steps smaller than one semitone, similarly to what can occur in a detuned instrument. The bins_per_octave parameter of the pitch-shifting function was set to $72 = 12 \times 6$ while the n_steps parameter was set to a random value between 1 and 5 for each sound. Neither 0 or 6 were selected as possible values as it would be the same as not altering the sound or pitch-shifting it by one semitone. The intention of the random assignment in the n_steps is to ensure the size of this augmented dataset is equal to the size of the datasets of other effects.

[28] https://tal-software.com/products/tal-dub
[29] https://soundhack.com/freeware
[30] https://bluecataudio.com/Products/Product_Flanger
[31] https://tal-software.com/products/tal-chorus-lx
[32] https://bluecataudio.com/Products/Product_Chorus

The audio resulting from this augmentation step can be longer than the original unprocessed audio. In order to keep all examples with the same length, the processed audio files were trimmed, ensuring all audio samples had a fixed duration of 4 s, similar to the sounds presented in the NSynth dataset(Engel et al., 2017).

The next step in the data processing pipeline is representing each sound in a log-scaled mel-spectogram. First, a 1024-point Short-Time Fourier Transform (STFT) is calculated on the signal, with a 75% overlap. The magnitude of the STFT result is converted to a mel-spectogram with 80 components, covering a frequency range from 40 Hz to 7600 Hz. Finally, the logarithm of the mel-spectogram is calculated, resulting in a 80×247 log-scaled mel-spectogram for the 4 s sounds sampled at 16 kHz present in the NSynth dataset (Engel et al., 2017).

4.2.2 Convolutional Neural Network

The CNN architecture we chose to use in our experiment is the *single-layer* architecture proposed by Pons et al. (2017b) for the musical instrument classification experiment, which has an implementation available online[33]. This architecture uses vertical convolution filters in order to better model timbral characteristics present in the spectogram, achieving close to state-of-the-art results of Han et al. (2017), using a much smaller model (23 times less trainable parameters) and a consequently lower training time.

We chose the *single-layer* architecture presented in this study and adapted it to take an input of size 80×247. This architecture contains a single but wide convolutional layer with different filters with various sizes, to capture the timbral characteristics of the input:

- 128 filters of size 5×1 and 8×1;
- 64 filters of size 5×3 and 80×3;
- 32 filters of size 5×5 and 80×5.

Batch normalization Ioffe & Szegedy (2015) is used after the convolutional layer and the activation function used is Exponential Linear Unit (ELU) (Clevert et al., 2016). Max pooling is applied in the channel dimension for learning pitch invariant representations. Finally, 50% dropout is applied to the output layer, which is a densely connected 11-way layer, with the *softmax* activation function. A graph of the model can be seen in Figure 4.1. For more information on this architecture and its properties see Pons et al. (2017b).

[33] https://github.com/Veleslavia/EUSIPCO2017

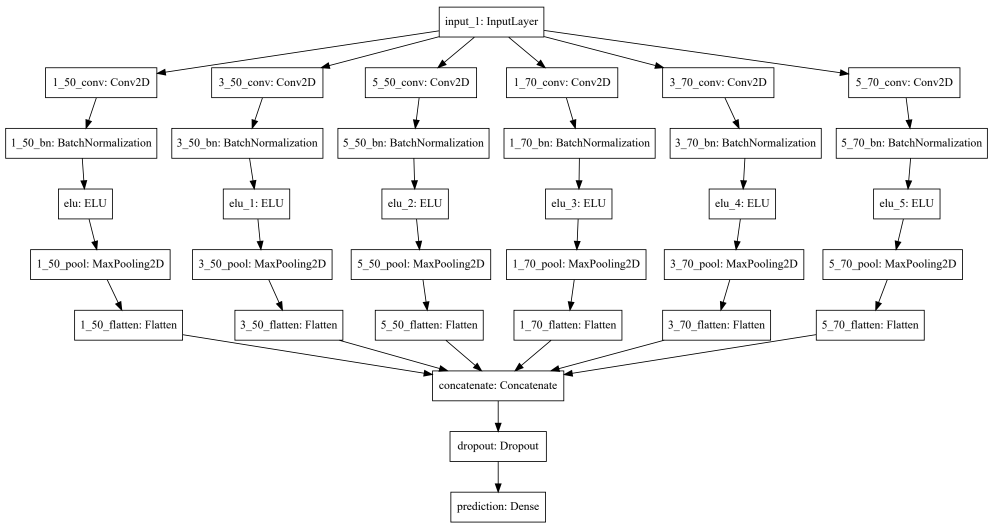


Figure 4.1: Single-layer CNN architecture proposed in Pons et al. (2017b)

4.3 Evaluation

The training of the models used the Adam optimizer (Kingma & Ba, 2015), with a learning rate of 0.001. In the original paper (Pons et al., 2017b) the authors used Stochastic Gradient Descent (SGD) with a learning rate reduction every 5 epochs. This was shown to provide good accuracy on the IRMAS dataset. However, we chose to use Adam as an optimizer because it does not need significant tuning as SGD. Furthermore, using a variable learning rate dependent on the number of epochs could benefit the larger training datasets as is the case of the ones with augmentation. A batch size of 50 examples was used, as it was the largest batch size able to fit the memory of the available GPUs. The loss function employed for the training was the categorical cross-entropy, as used in Pons et al. (2017b), which can be calculated as shown in Equation (4.1), where N represents the number of observations (examples in the training set) and $p_{model}[y_i \in C_{y_i}]$ is the predicted probability of the i^{th} observation belonging to the correct class C_{y_i}.

$$loss = -\frac{1}{N}\sum_{i=1}^{N} \log p_{model}[y_i \in C_{y_i}] \tag{4.1}$$

To compare the models trained with the different datasets, we used categorical accuracy as evaluation metric, described in Equation (4.2). A prediction is considered correct if the index of the output node with highest value is the same as the correct label.

$$\text{Categorical Accuracy} = \text{Correct predictions/N} \tag{4.2}$$

All the models were trained until the categorical accuracy did not improve in the validation set after 10 epochs and the model which provided the best value for the validation set was evaluated in the test set.

4.4 Results

Two experiments were conducted in our study. We firstly evaluated how augmenting the training set of NSynth by applying audio effects to the sounds can improve the automatic classification on the instruments of the unmodified test set. In the second experiment we evaluated how robust a state-of-the-art model for instrument classification is when classifying sounds where these audio effects are applied.

Test Effect	Train Effect	Accuracy
None	None (baseline)	0.7378
	Heavy distortion	**0.7473**
	Saturation	0.7349
	Reverb	0.7375
	Chorus	**0.7417**
	Echo	0.7336
	Flanger	**0.7412**
	Pitch Shifting	0.7334

Table 4.1: Classification accuracy on the unprocessed test set.

The results of the first experiment are presented in Table 4.1, where the classification accuracy between the models trained with the original NSynth training set augmented with audio effects can be compared to the baseline (unprocessed dataset). We see that the increase in accuracy only occurs for chorus, heavy distortion and flanger effects. The highest classification accuracy was achieved by the dataset augmented with heavy distortion, where an increase of 1% was obtained. However, all the accuracy values are in a small interval (between 0.7334 and 0.7473), which means that the model was not able to learn from the augmented datasets. Future experiments are needed in order to understand why this occurs. In Salamon & Bello (2017), the authors state that the superior performance obtained was due to an augmentation procedure coupled with an increase in the model capacity. Experiments with higher capacity models can be performed to understand if the size of the model used is limiting its performance on learning from the augmented dataset.

In Table 4.2, we present the accuracy values obtained when evaluating the trained model on test sets processed with effects. The first thing we verify is that the accuracy of the classification greatly decreases for almost all effects,

Test Effect	Train Effect	Accuracy
Heavy distortion	None	0.3145
	Heavy distortion	**0.3518**
Saturation	None	**0.4836**
	Saturation	0.4607
Reverb	None	**0.3931**
	Reverb	0.3774
Chorus	None	0.6348
	Chorus	**0.6436**
Echo	None	**0.4719**
	Echo	0.4319
Flanger	None	**0.7046**
	Flanger	0.7002
Pitch Shifting	None	**0.6980**
	Pitch Shifting	0.6741

Table 4.2: Classification accuracy on the augmented test set.

when compared to the unprocessed sound classification. The model seems to be more robust to the flanger and to the pitch shifting effect, where the difference between the accuracy on the unprocessed test set and on the processed one is smaller than 4%. The effects which caused the biggest drops in accuracy ($>$ 20%) were the heavy distortion, the saturation, the echo and the reverb. When evaluating if training with the augmented datasets increased the robustness of the model, we see that this is only true for the chorus and distortion effect. While for the heavy distortion effect the accuracy when training with the augmented set is improved by a significant value ($\approx$ 4%), the difference in accuracy between training with the augmented and the unprocessed sets are generally small. Further experiments can be performed to understand the bad generalization of the model.

4.5 Conclusions

In this chapter, we evaluated how a state-of-the-art algorithm for automatic instrument classification performs when classifying the NSynth dataset and how augmenting this dataset with audio effects commonly used in electronic music production influences its accuracy on both the original and processed versions of the audio. We identify that the accuracy of this algorithm is greatly decreased when tested on sounds where audio effects are applied and see that the augmentation can lead to better classification in unprocessed sounds.

Future work includes experimenting with a higher capacity model. Work can also be conducted on further augmenting the datasets. Although the effects applied were the same in the training, validation and test sets, the implementations used was different in the training set. This leads to a different timbre between the sets that the architecture might not be able to generalize to. Augmenting the dataset using a number of different settings for each effect, as well as different combinations of the effects applied are other future possibilities for research.

CHAPTER 5

Freesound Loop Dataset and Annotation Tool

Music loops are essential ingredients in electronic music production, and there is a high demand for pre-recorded loops in a variety of styles. Several commercial and community databases have been created to meet this demand, but most are not suitable for research due to their strict licensing. In this chapter, we present the Freesound Loop Dataset (FSLD), a new large-scale dataset of music loops annotated by experts. The loops originate from Freesound, a community database of audio recordings released under Creative Commons licenses, so the audio in our dataset may be redistributed. The annotations include instrument, tempo, meter, key and genre tags. We describe the methodology used to assemble and annotate the data, and report on the distribution of tags in the data and inter-annotator agreement. We also present to the community an online loop annotator tool that we developed. To illustrate the usefulness of FSLD, we present short case studies on using it to estimate tempo and key, generate music tracks, and evaluate a loop separation algorithm. We anticipate that the community will find yet more uses for the data, in applications from automatic loop characterization to algorithmic composition.

This chapter is based on **Ramires, A.**, Font, F., Bogdanov, D., Smith, J., Yang, Y.H., Ching, J., Chen, B.Y., Wu, Y.K., Wei-Han, H., & Serra, X. (2020). The Freesound Loop Dataset and Annotation Tool. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR).*

5.1 Introduction

Audio loops have been made available for amateur and professional music makers since the early ages of electronic music. Currently, large-scale databases of audio offer huge collections of audio material for users to work with. Some databases, like Freesound[34] and Looperman[35], are community-oriented: people upload their sounds so that other users can employ them in their works. More commonly, collections are commercially oriented: loops are available to paying costumers, either through a subscription service (*e.g.* Sounds.com[36], Splice[37]) or by allowing customers to buy packs of loops (*e.g.* Loopmasters[38], and Prime Loops[39]).

Despite the number of loops available on these databases, the technologies used to analyse and navigate these databases still rely on human annotations and human content curation to, for instance, group sounds into packs for specific genres or styles. Loops are being manually annotated with information like instrument, tonality (key), tempo (BPM) and music genre. This is a time-consuming task which is often unfeasible, which results in badly annotated databases and poor user experience when browsing them. In the field of Music Information Retrieval (MIR), a substantial effort has been put into automatically identifying the aforementioned characteristics for musical pieces. However, loops are inherently different from music pieces (*i.e.* with reduced instrumentation and short length). Therefore, existing MIR algorithms need to be tested and (possibly) adapted to work successfully in this scenario.

Early work on the retrieval of loops focused on tempo extraction and transcription from drum loops (Gillet & Richard, 2004; Gouyon et al., 2006). Gouyon et al. (2006) compared several tempo induction algorithms proposed in the ISMIR 2004 competition. The loop dataset used in this work has been commonly used for evaluating tempo estimation algorithms and is divided into three subsets. One of these comprises two thousand audio loops (with tempo annotations) from Sound Effects Library[40]. These audio loops are not free and a license needs to be obtained to use them for research.

Automatic transcription of drum loops focuses on identifying when the different percussion instruments occur in a loop. Gillet & Richard (2004) used a collection of 315 drum loops for evaluating their system and provided "a compressed version of a few drum loops". The URL to the webpage the authors provide is broken and, presumably, the lower quality versions of the loops do not represent commercial-quality content. The authors also use this dataset for automatically retrieving drum loops from spoken queries (Gillet & Richard, 2005). This database was later used by Ravelli et al. (2007) and Bello et al.

[34]https://freesound.org
[35]https://looperman.com
[36]https://sounds.com
[37]https://splice.com
[38]https://loopmasters.com
[39]https://primeloops.com
[40]http://sound-effects-library.com

(2006) for automatic rhythm modification and analysis of drum loops.

The work of Gómez-Marín et al. (2015) explores rhythmic similarity measures for audio loops . The authors validate the proposed metric using 9 drum break loops from Rhythm Lab[41]. The authors do not specify which are the drum loops used.

Font & Serra (2016) presented a dataset of audio loops from Freesound (Font et al., 2013) in their work on tempo estimation and a confidence measure for tempo in audio loops. The authors use two commercial datasets, loops bundled with music production software Apple's Logic Pro[42] and Acoustica's Mixcraft[43], and two community datasets. The first one is a private collection of loops downloaded from Looperman, which was previously used for research in Roma (2015). Looperman does not allow the re-distribution of loops "as is", and considers as misuse the automatic download of their loops[44]. A collection of 4000 loops from Freesound, obtained by searching Freesound for sounds with the queries "loop" and "bpm" is also proposed. The sounds' filenames, tags and textual descriptions are parsed to identify tempo annotations provided by the users. However, these annotations are not always accurate, and, to enable further work on audio loops, more information besides the tempo is desired.

In short, existing academic work which employs loops resorts to commercial samples as the source of data and open datasets do not have complete and reliable annotations. This makes it difficult to reproduce existing research. To promote open and accessible research on audio loops, we propose a free and distributable database of loops from Freesound, which provides production-ready sounds with high-quality annotations.

We present the Freesound Loop Dataset (FSLD), an open dataset with 9,455 music loops to support reproducible research in MIR. FSLD contains production-ready loops from Freesound which are distributed under Creative Commons licenses and can, therefore, be freely shared among the research community and industry. Part of the dataset has been manually annotated with information about rhythm, tonality, instrumentation and genre, in a similar way as commercially available loop collections are annotated. The annotation service is made public[45] so that the community can work on enlarging the annotations of this collection. We expect this dataset to have an impact on the research community as it supports further research into several timely research topics which are also of great interest to the industry.

The rest of the chapter is structured as follows. Section 5.2 details how the

[41]https://rhythm-lab.com
[42]https://apple.com/logic-pro
[43]https://acoustica.com/mixcraft
[44]https://looperman.com/help/terms
[45]http://mtg.upf.edu/fslannotator

proposed dataset was collected and annotated. In Section 5.3, general statistics of the dataset are given. In Section 5.4 and 5.5, we present some potential applications and provide a benchmark of the dataset using some classic MIR tasks. Finally, in Section 5.6, we conclude and suggest future work directions.

5.2 Dataset Creation

In this section the process we have followed to create the dataset is described. We show how we collected the loops to annotate, how they were pre-analyzed for a faster annotation procedure and explain what was annotated and how the annotation tool was implemented. Finally, we present how the dataset is distributed and organized.

5.2.1 Loop Selection

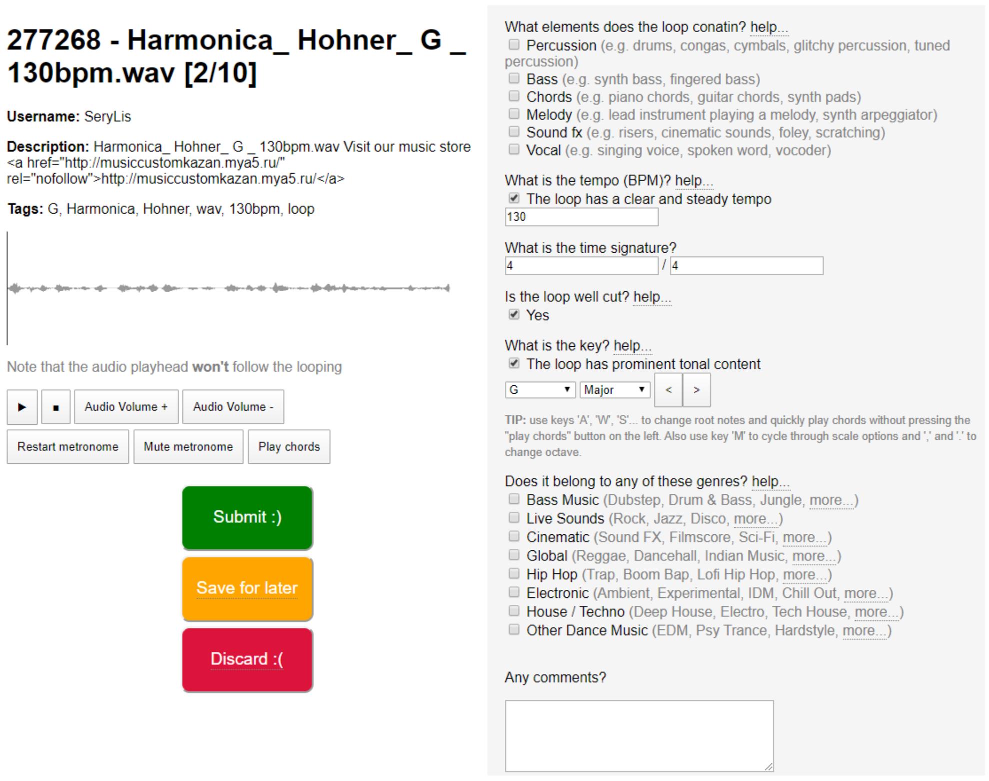
To select an initial pool of candidate loops, we followed the same methodology as in Font & Serra (2016): i.e., we retrieved sounds with both "loop" and "bpm" keywords on Freesound, resulting in 9,490 sounds. Using the Freesound API, it was straightforward to obtain these loops and their metadata—title, tags, textual description, and author's username.

5.2.2 Loop Annotation

We want the loops in our dataset to be annotated in a way which is similar to commercially available loops. This way, we make sure that the loop characterization is compatible with industry standards. For this, we decided to annotate the loops' instrumentation, tempo, time signature, key and genre, as described below. The annotation was performed by 8 MIR researchers and students, with knowledge of electronic music production. To make the annotation procedure as efficient as possible, we created a web application for the annotators with several tools at their disposal, which can be seen in Fig. 5.1. This application was developed using Flask[46], a web framework for Python.

This interface provides fields for the annotators to fill in the desired information, which will be described in the following sections. The instructions are provided on tooltips for quick access by annotators.

[46] https://flask.palletsprojects.com



277268 - Harmonica_ Hohner_ G _ 130bpm.wav [2/10]

Username: SeryLis

Description: Harmonica_ Hohner_ G _ 130bpm.wav Visit our music store <a href="http://musiccustomkazan.mya5.ru/" rel="nofollow">http://musiccustomkazan.mya5.ru/</a>

Tags: G, Harmonica, Hohner, wav, 130bpm, loop

Note that the audio playhead **won't** follow the looping

What elements does the loop conatin? help...
- ☐ Percussion (e.g. drums, congas, cymbals, glitchy percussion, tuned percussion)
- ☐ Bass (e.g. synth bass, fingered bass)
- ☐ Chords (e.g. piano chords, guitar chords, synth pads)
- ☐ Melody (e.g. lead instrument playing a melody, synth arpeggiator)
- ☐ Sound fx (e.g. risers, cinematic sounds, foley, scratching)
- ☐ Vocal (e.g. singing voice, spoken word, vocoder)

What is the tempo (BPM)? help...
- ☑ The loop has a clear and steady tempo

130

What is the time signature?

4 / 4

Is the loop well cut? help...
- ☑ Yes

What is the key? help...
- ☑ The loop has prominent tonal content

G Major

TIP: use keys 'A', 'W', 'S'... to change root notes and quickly play chords without pressing the "play chords" button on the left. Also use key 'M' to cycle through scale options and ',' and '.' to change octave.

Does it belong to any of these genres? help...
- ☐ Bass Music (Dubstep, Drum & Bass, Jungle, more...)
- ☐ Live Sounds (Rock, Jazz, Disco, more...)
- ☐ Cinematic (Sound FX, Filmscore, Sci-Fi, more...)
- ☐ Global (Reggae, Dancehall, Indian Music, more...)
- ☐ Hip Hop (Trap, Boom Bap, Lofi Hip Hop, more...)
- ☐ Electronic (Ambient, Experimental, IDM, Chill Out, more...)
- ☐ House / Techno (Deep House, Electro, Tech House, more...)
- ☐ Other Dance Music (EDM, Psy Trance, Hardstyle, more...)

Any comments?

Figure 5.1: The user interface provided to the annotators, available online[45].

5.2.2.1 Instrumentation

Instead of annotating instruments in a traditional way, which would not be straightforward in heavily processed audio or more experimental loops, we chose to annotate general *roles* which can be useful for both music makers and automatic generation of music. We asked annotators to tick all the roles that apply to each loop. Usually, specific instruments could be easily assigned to a specific role. We present the roles along with some examples in Table 5.1.

Role	**Example Instruments**
Percussion	Drums, glitches, tuned percussion
Bass	Synth bass, fingered bass
Chords	Piano chords, guitar chords, synth pads
Melody	Instrument playing a melody, arpeggiator
Sound FX	Risers, cinematic sounds, foley, scratching
Vocal	Singing voice, spoken word, vocoder

Table 5.1: Instrumentation roles and the examples provided for each category.

5.2.2.2 Rhythmic Characteristics

We asked for annotations on three rhythmic aspects:

Tempo provides an easy measure of rhythmic compatibility and is the most common information provided in commercial loop databases. We ask annotators if the loop has a clear and steady tempo, to identify loops with constant tempo and clear beat (BPM value and steady tempo), with changing tempo (BPM value of the initial tempo and no steady tempo), and loops with no clear beat but where the tempo can be inferred (BPM value and no steady tempo).

Meter is not a feature we see annotated as often as BPM, which might be due to the common use of 4/4 meter in electronic music. This feature is relevant to annotate, for calculating the number of bars in a loop, from its meter, tempo and duration.

Finally, as sometimes the length of the audio file is not the length of the loop, we also annotate if it is **well-cut**. If there is some silence at the beginning or the end of the file or if there is a "tail" (*e.g.* a decay of a reverb effect) when the audio is exported, it might not loop correctly just by staring the loop again when it finishes playing.

5.2.2.3 Tonal Characteristics

We annotate if the loop has prominent tonal content and, if so, to indicate a root key and mode that matches the tonal content of the loop (*i.e.*, root note from a chromatic scale and *Major*/*Minor* mode). We explained "prominent tonal content" as whether it is easy to sing along to the loop or to find a meaningful root note for the loop. For root key annotations, we asked to choose a note from a dropdown with 12 notes, or "Unknown" in case the key could not be found. For annotating mode, the annotators had the choice of "Major" or "Minor" if the loop sounded good with one of these modes; "None" if the loop could not be clearly assigned to either "Major" or "Minor" (e.g. loop contains a single note); or "Unknown" for other cases.

5.2.2.4 Genre

We annotate genre in non-exclusive categories, where each is assigned to a loop if it can be used to make music in that genre. To create a taxonomy which would be similar to commercially available ones, we merged the taxonomies of Sounds.com and Splice. These were chosen as they provided several examples for each genre and had similar parent categories. We present the taxonomy in Table 5.2.

Genre	Examples
Bass Music	Dubstep, Drum and Bass, Jungle
Live Sounds	Rock, Jazz, Disco
Cinematic	Sound FX, Filmscore, Sci-Fi
Global	Reggae, Dancehall, Indian Music
Hip Hop	Trap, Boom Bap, Lofi Hip Hop
Electronic	Ambient, IDM, Chill Out
House / Techno	Deep House, Electro, Tech House
Other Dance Music	EDM, Psy Trance, Hardstyle

Table 5.2: Taxonomy of genres used for the annotation and examples for each category.

5.2.2.5 Loop Pre-Analysis and Annotation Tools

We performed a pre-analysis on the loops to obtain tempo, key and genre suggestions. To obtain the tempo information, we followed the same approach of Font & Serra (2016), parsing the title, description and the tags of the loop for tempo information provided by users. To propose an initial key and mode to the annotators, we analyzed the loops using the algorithm proposed by Faraldo et al. (2016), which is implemented in the Essentia audio analysis library (Bogdanov et al., 2013). Finally, by taking the genre information from the textual metadata of the loops, we were able to map some of the sounds to the genres to annotate. The checkboxes were selected for the genres which either were mentioned or had a sub-genre mentioned in the textual metadata. Our annotators were familiar with the annotation procedure and took the pre-annotations only as suggestions to speed-up the annotation process.

In the annotation tool, at the top of the display is the loop's metadata: its unique sound id, title, author's username, and the tags and textual description provided by the author (see Fig. 5.1). The waveform of the loop and a playhead is also shown, which are linked to an audio player. The audio player always restarts the playback of the loop when it finishes, and triggers a metronome with the BPM provided in the BPM annotation field. We provide stop, play and pause controls for the loop and metronome and volume controls for the loop. A button which restarts only the metronome is also present. To ease finding a key and mode which suits the loop, we present a synthesizer which plays the chord present in the tonal annotation section. In case the mode selected is "None" or "Unknown", the synthesizer will just play the root note of the key selected. Using the computer's keyboard, the annotator can cycle through the options for key and mode, in several octaves. Finally, buttons are provided for submitting the annotation when it is finished, saving the sound for later and discarding the sound in case it is not a loop.

5.2.3 Dataset Availability

The loops and corresponding annotations (provided in a JSON file) are publicly available on Zenodo[47]. This dataset can be divided into three subsets, defined by their level of annotations. These are:

- Multiple-annotations (MA): the loops annotated by at least two researchers. It contains 1,472 loops.
- Single-annotation (SA): the loops annotated by a single researcher. Currently contains 1,464 loops.
- Automatic-annotations (AA): the loops annotated by the analysis algorithms mentioned in Section 5.2.2.5. Contains 9,455 loops: the loops in MA and SA and 6,519 more.

In addition to the main dataset, we provide a repository[48] with the code used for the annotation tool interface and server, the pre-analysis that generated the subset of automatic annotations, and the analysis and potential applications presented in Sections 5.3, 5.4 and 5.5.

5.3 Dataset Analysis

To understand the diversity and reliability of the dataset, we investigate the distribution of annotated characteristics and inter-annotator agreement.

5.3.1 Annotation Distribution

The human-annotated part of the dataset contains 1,579 sounds which, in total, have been annotated 2,809 times. The distribution of genres, instrumentation, and keys are shown in Tables 5.3 and 5.4 and the tempo histogram in Figure 5.2. It is well-balanced in terms of instrument and genre; reasonably balanced in terms of tempo, although 120 bpm dominates; and highly imbalanced in terms of key, with C Major and Minor dominating.

5.3.2 Inter-annotator Agreement

To measure the agreement of the annotators in our dataset, we measure the inter-annotator agreement for the MA annotations subset. To do this, we use

[47]https://doi.org/10.5281/zenodo.3967852
[48]https://github.com/aframires/freesound-loop-annotator

Percussion	54.95%
Bass	19.10%
Chords	11.90%
Melody	21.31%
FX	24.80%
Vocal	2.29%

Bass Music	32.04%
Live Sounds	21.38%
Cinematic	19.95%
Global	14.26%
Hip-hop	17.29%
House/Techno	29.05%
Other Dance Music	25.63%

Table 5.3: Distribution of the instrument roles and genre in our dataset.

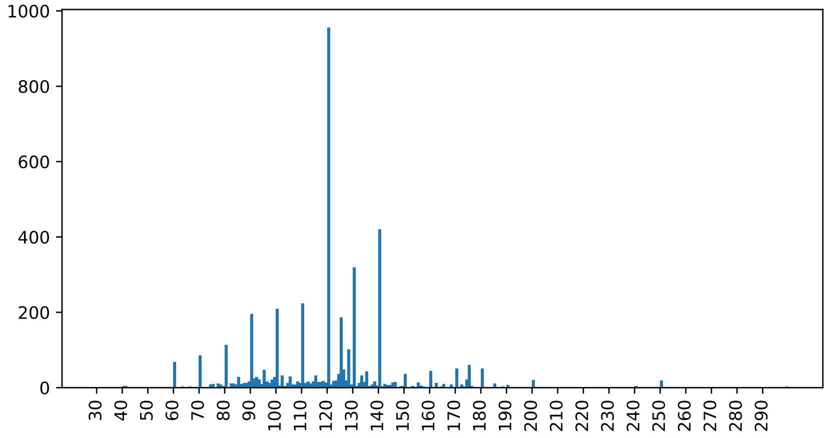


Figure 5.2: Distribution of BPMs in FSLD.

two metrics: proportion of overall agreement (Agr.) for all the annotations, and positive and negative agreement (PA and NA) (Feinstein & Cicchetti, 1990) for binary classification tasks. The proportion of overall agreement reflects the number of cases when both annotators agree on a label, and is calculated by dividing their number by the total number of annotations. This overall metric does not distinguish the agreement in positive and negative cases, so for the binary annotation tasks we also calculated the positive and negative agreement. The formulas for calculating these are given in Eq. 5.1, where the variables represent the annotations by the annotators (e.g., NP = first annotator answered negative, second positive).

$$PA = \frac{2PP}{2PP+NP+PN}, \tag{5.1}$$

$$NA = \frac{2NN}{2NN+NP+PN}, \tag{5.2}$$

Table 5.5 presents the results for this analysis. We can see that overall, the values for the agreement are high. Bass, melody and chords have a lower positive agreement value, despite the high negative agreement. This might

Key	**Maj**	**Min**	**None**	**Unknown**
C	9.63%	8.38%	3.65%	0.95%
C#	1.38%	2.84%	0.60%	0.43%
D	3.31%	5.37%	1.85%	0.39%
D#	1.29%	2.49%	0.73%	0.21%
E	2.28%	3.65%	1.20%	0.26%
F	4.64%	4.43%	1.16%	0.34%
F#	1.12%	2.49%	1.12%	0.13%
G	2.66%	4.25%	1.93%	0.43%
G#	1.72%	2.58%	0.90%	0.13%
A	3.01%	5.50%	1.68%	0.26%
A#	1.50%	1.89%	0.52%	0.04%
B	0.82%	3.01%	0.64%	0.21%

Table 5.4: Distribution of the keys in our dataset.

Char.	**Sub-Char.**	**Agr.**	**PA**	**NA**
Inst.	Percussion	85.16%	86.62%	83.35%
	Bass	76.73%	45.83%	85.19%
	Melody	82.33%	60.57%	88.61%
	Chords	87.40%	47.35%	92.84%
	FX	72.04%	43.61%	81.41%
	Vocal	98.66%	71.88%	99.31%
Tempo	BPM	87.84%	NA	NA
	Signature	97.84%	NA	NA
	Well Cut	86.88%	92.73%	32.82%
Key	Root	67.56%	NA	NA
	Mode	69.80%	NA	NA
Genre	Bass Music	69.50%	53.26%	77.37%
	Live Sounds	80.09%	55.28%	87.19%
	Cinematic	81.66%	57.14%	88.33%
	Global	82.33%	51.53%	89.19%
	Hip-Hop	79.05%	31.30%	87.64%
	House/Techno	69.35%	48.56%	78.17%
	Other	73.53%	45.64%	82.50%

Table 5.5: Inter-annotator agreement for the MA subset.

indicate that annotators are not able to easily distinguish if an element should fit in one of the 3 roles, but can say when it is not present. The lower value for root key agreement indicates that several keys are used to describe the same sounds. This fits our annotating indications, where we asked annotators to select a key which sounds good with the loop and therefore, personal taste may arise in this choice. Finally, the positive agreement for genres always has values lower than 65%, which might be due to how genre might be perceived subjectively between annotators.

5.4 Benchmarking MIR Tasks

To demonstrate the usefulness of this dataset, we use it in several short case studies. To benchmark tempo, we followed the evaluation approach of Font & Serra (2016) and used the *Accuracy 1* and *Accuracy 2* presented in Gouyon et al. (2006), together with the *Accuracy 1e* proposed in Font & Serra (2016). Due to space constraints, here we only report the mean of the 3 accuracies. Full results can be seen in an accompanying website[49]. The algorithms selected for the tempo benchmarking were the following (details for each algorithm can be found in respective papers):

- **Percival (Percival & Tzanetakis, 2014):** We use both the original implementation and the one provided in Essentia.
- **Zapata (Zapata et al., 2014):** Implementation provided in Essentia.
- **Degara (Degara et al., 2012):** We also use Essentia's implementation.
- **Böck (Böck et al., 2015):**We use the 3 variants available in the Madmom library[50]: COMB, ACF and DBN.

We validate tempo estimation algorithms on the 3 proposed subsets. Key estimation is only validated on the MA and SA subsets as we do not have original uploader annotations for key. The MA subset, which has at least 2 annotations per loop, was analyzed in two ways: BOTH and EITHER. In BOTH, we run the MIR algorithms exclusively on the loops which have the same labels from both annotators. In EITHER, the output of the algorithm was deemed correct if it was at least one of the annotated labels. The results are presented in Table 5.6

We can see that the results are similar to the ones obtained in Font & Serra (2016), with Percival14 having better accuracy across all the datasets. We can see that the accuracy increases from AA to SA, and from SA to BOTH. This might be due to the user-annotated loops having incorrect annotations; it may also be that when both annotators agree on a tempo, the tempo is strong and defined. The EITHER evaluation gives the largest accuracies, which may be due to its broader criteria for considering tempos correct.

[49]https://aframires.github.io/freesound-loop-annotator
[50]https://github.com/CPJKU/madmom

Algorithm	AA	SA	BOTH	EITHER
Percival14	58.09	62.98	65.75	84.13
Percival14e	57.82	64.00	65.49	84.98
Zapata14	51.81	58.79	58.99	77.97
Degara12	52.32	58.77	59.31	79.16
Bock15COMB	44.42	51.17	52.92	71.35
Bock15ACF	48.65	51.96	54.75	74.90
Bock15DBN	45.76	50.60	52.32	70.90

Table 5.6: Evaluation of tempo estimation algorithms in the proposed subsets.

For benchmarking key estimation algorithms, we used the evaluation metrics from MIREX[51], which evaluates how *close* the estimated key and the annotated key are to provide an accuracy. The algorithms compared in the evaluation were the following:

- **EDMKey (Faraldo et al., 2016):** We use the implementation in Essentia, with 4 key profiles: Krumhansl (Krumhans, 1990), Temperley (Temperley, 1999), Shaath (Sha'ath, 2011) and the one proposed in (Faraldo et al., 2016).
- **EssentiaBasic (Bogdanov et al., 2013):** Essentia's implementation of the algorithm presented by Gómez (Gómez, 2006b).
- **QMUL (Noland & Sandler, 2007):** We use the Key Detection implementation available in QM Vamp Plugins[52].

In Table 5.7, we present part of the results of the key estimation evaluation. Due to lack of space, only the final MIREX scores for each dataset are presented. The full results can be seen in the accompanying website[49].

Algorithm	SA	BOTH	EITHER
Edmkey	72.26	88.25	85.63
EdmkeyKrumhansl	66.99	84.85	82.98
EdmkeyTemperley	61.46	71.78	71.77
EdmkeyShaath	72.38	88.25	85.63
EssentiaBasic	71.25	88.80	85.30
QMULKeyDetector	35.09	42.15	46.25

Table 5.7: Evaluation of key estimation algorithms in the proposed subsets.

We see that EssentiaBasic and EDMkey are the best performing algorithms here. EDMKey has been specially tuned to be used for EDM, which might

[51] https://music-ir.org/mirex/wiki/2019:Audio_Key_Detection
[52] https://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-keydetector

make it more suitable to the loops we are annotating. We again see that the accuracy increases from SA to BOTH, which might indicate again that when the key is clear and defined, the algorithms are also able to correctly identify it.

5.5 Music Generation and Decomposition

Another way the dataset is valuable is for creating synthetic datasets of songs for evaluating loop-extraction algorithms, such as Smith & Goto (2018). We created 100 random songs, each using 5 random drum loops and 5 non-drum loops (chosen from a subset of 4/4, 120-bpm, 1-bar, single-instrument loops for which there was no disagreement among the annotators on the instrument role). Each song is a random arrangement of the loops, either in a *sparse* arrangement, in which one drum and one non-drum loop occurs per bar, or a *dense* one, in which 4 loops occur per bar (i.e., 2 drum and 2 non-drum). For comparison, we also recreated the "composed" and "factorial" layouts from Smith & Goto (2018). Examples of each layout are shown on the accompanying website.

We used the public implementation[53] of Smith & Goto (2018) to extract loops for each song, informed with the true number of loop segments (4 or 10) and the true downbeat boundaries. The metrics SDR, SIR and SAR (the signal to distortion, interference and artefacts ratios (Raffel et al., 2014)) are reported in the left part of Table 5.8.

These are normally computed by trying all permutations of estimated sources to true sources and using that which maximizes the score. This is infeasible for permutations of 10 items, so we first find the permutation that maximizes the similarity between the source and true loop spectra.

Layout	**SDR**	**SIR**	**SAR**	**F1**	**Acc.**
Sparse	–5.2	–3.9	15.8	0.194	0.691
Dense	–7.9	–7.3	14.4	0.294	0.542
Composed	12.5	18.4	22.6	0.585	0.546
Factorial	19.8	29.2	24.1	0.560	0.551

Table 5.8: Evaluation of loop source quality (SDR, SIR, SAR) and estimated layouts (F-measure and accuracy) for each song layout.

This permutation is also used to evaluate the quality of the estimated layout. We binarize each row of the estimated layout, using the row's mean as threshold. We then compute the raw accuracy (as in Smith & Goto (2018)),

[53] https://github.com/jblsmith/loopextractor

but here we propose also using the F-measure, so as not to weight true negatives unduly. The results are in the right columns of Table 5.8.

SDR, SIR and SAR are all lower for the random 10-part songs than for the 4-part songs, showing that we have created a more challenging testing ground for loop extraction systems. For the layout evaluation, our evaluation makes clear that the raw accuracy gives undue weight to true negatives: the highest accuracy was obtained for the sparse layouts, despite having the lowest F-measure. This short evaluation is a proof of concept; with more space, we could study the impact of the instrumentation, number of loops, loop duration, and other factors on the separation quality. We can also generate layouts with loops of many durations and evaluate hierarchical loop extraction systems.

5.6 Conclusion

In this chapter, we presented our work on addressing the lack of standard loop datasets to carry MIR tasks. We presented FSLD, a dataset of audio loops annotated at a level similar to commercial loop collections. These loops are licensed for redistribution and can be used and redistributed for research purposes. We provide a detailed analysis of the dataset and its annotations and provided several use cases for tempo and key benchmarking, music generation and loop separation. Furthermore, we present the online annotation tool used to build the dataset, and we make it available online so other researchers and the general public can contribute and extend the dataset.

CHAPTER 6

Automatic Instrumentation Role Classification

In this chapter, we look at labeling loops of key structural groups such as bass, percussive or melodic elements by the role they occupy in a piece of music through the task of Automatic Instrumentation Role Classification (AIRC). Such labels assist Electronic Music (EM) producers in the identification of compatible loops in large unstructured audio databases. While human annotation is often laborious, automatic classification allows for fast and scalable generation of these labels. We experiment with several deep-learning architectures and propose a data augmentation method for improving multi-label representation to balance classes within the Freesound Loop Dataset (FSLD). To improve the classification accuracy of the architectures, we also evaluate different pooling operations and, additionally, we demonstrate how our proposed AIRC method is useful for analyzing the structure of EM compositions through loop activation transcription.

This chapter is based on the early experiments conducted in Ching, J., **Ramires, A.**, & Yang, Y.H. (2020). Instrument Role Classification: Auto-tagging for Loop Based Music. In *Proceedings of The 2020 Joint Conference on AI Music Creativity (MuMe + CSMC)*, while most of the contributions are taken from Drysdale, J., **Ramires, A.**, Serra, X., & Hockman, J. (2022). Improved Automatic Instrumentation Role Classification and Loop Activation Transcription. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

6.1 Introduction

Loops serve as the material from which music makers can generate EM compositions through various editing and combinatory processes (e.g., layering, splicing, rearranging). Figure 6.1 depicts a simplified representation of the EM creation process involving the layering and repeated activation of loops with different roles.

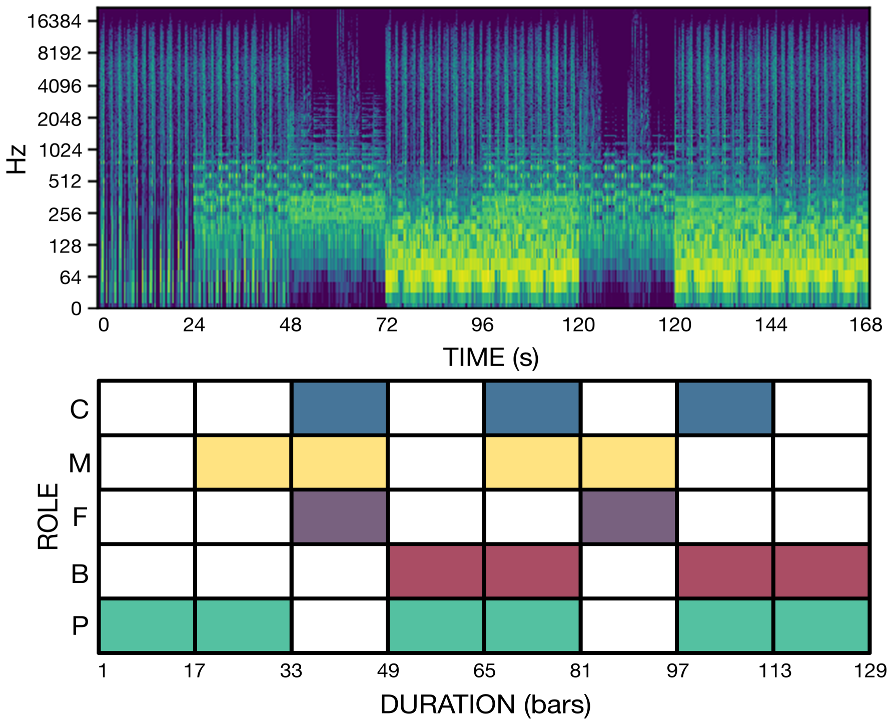


Figure 6.1: A simplified EM composition structure, built with five loop layers. Log-scaled STFT power spectrogram (*top*) and corresponding role activations: Chords (C), Melody (M), Sound Fx (F), Bass (B), and Percussion (P) at 4-bar intervals (*bottom*).

In recent years, there has been an increased focus on research related to audio loops within the field of music information retrieval. There are several methods that exist for automated loop retrieval (Gillet & Richard, 2005; López-Serrano et al., 2017; Shi & Mysore, 2018; Smith et al., 2019; Chen et al., 2020), and loop creation (Cocharro et al., 2014; Alain et al., 2020; Chandna et al., 2021).

In addition, there are two recent methods proposed for loop activation transcription, a task that involves estimating the locations in which loops occur throughout a piece of music. López-Serrano et al. (2016) proposed a method for decomposing loop-based EM using Non-Negative Matrix Factorization Deconvolution (NMFD) (Smaragdis, 2004) to estimate spectral templates and rhythmic activations from magnitude spectrograms. Following this work, Smith & Goto (2018) propose an alternative method to discovering loop activations of EM using Non-Negative Tensor Factorization (NTF) (FitzGerald et al., 2006). While non-negative matrix factorization approaches allow for separation of mixed audio into the constituent loops of a music composition, they rely on non-varying repetitions of loops and do not optimize independence between learned loop representations.

As an alternative to the aforementioned approaches, which seek to identify the instrument within a loop and its associated activation, AIRC is a music auto-tagging task that estimates the presence of active instrumentation role groups (e.g., percussion, bass, melody, chords, sound Fx) within audio recordings. Research in AIRC has been facilitated by the development of the Freesound Loop Dataset (FSLD), presented in the previous chapter, due to its large amount of loops and corresponding instrumentation role annotations.

We apply AIRC to full EM compositions, in which multiple instrumentation roles (e.g., percussion, melody, bass) are often active. To handle the lack of co-occurring loops in the FSLD, we introduce a novel data augmentation technique to balance classes and experiment with several deep-learning architectures and pooling operations, resulting in a state-of-the-art AIRC system.

We then demonstrate the usefulness of AIRC in EM structural analysis by comparing our system with previous approaches for loop activation estimation. Additionally, the proposed AIRC system is shown to derive key structural information from full-length EM compositions in the form of instrumentation role activation maps, which would be of use in tasks such automatic DJing (Vande Veire & De Bie, 2018), mashups (Davies et al., 2013), and loop creation (Shi & Mysore, 2018).

The remainder of this chapter is structured as follows: Section 6.2, presents the proposed method for AIRC and loop activation transcription. Evaluation methodology and the datasets used in this study are detailed in Section 6.3 and the results and discussion are provided Section 6.4. Conclusions and suggestions for future work are presented in Section 6.5.

6.2 Methodology

In this study, several Convolutional Neural Network (CNN) architectures are evaluated to identify the best system for AIRC. Each architecture utilizes different configurations of front-end filter shapes to learn a representation from spectrograms and pooling operations that derive the final predictions by summarizing the information learned by the network.

As the data employed in AIRC contains different types of musical audio, from tonal melodies to noise-like sound Fx, this motivates the experimentation of architectures aimed at different sound classification tasks. Three front-end filter shapes are used: general domain square filters, vertical filters (Pons et al., 2017b) tailored towards classifying the timbre of melodic instruments, and harmonic band-passfilters which capture harmonic characteristics(Won et al., 2020).

To improve the AIRC predictions, two methods for summarizing the information learned in the final convolutional layers of a CNN are investigated. The standard approach to is to use Global Max Pooling (GMP); however, this infers strict assumptions about the label characteristics of the data. In the closely related field of sound event detection, auto-pooling has been proposed to automatically learn the best suited operation by interpolating between max-, mean-, or min-pooling during training. We implement both GMP and auto-pooling and compare their performance for the task of AIRC.

6.2.1 Implementation

Audio is input into the networks as a spectrogram representation, from which features are extracted through convolutional layers. Output predictions return values between $[0., 1.]$ depicting the presence of active instrumentation roles.

For each network, the input layer is a four-dimensional tensor $t \in \mathbb{R}^{b\times w\times h\times c}$, with batch size b, number of frames w, number of frequency bins h, and channels c. Following Pons et al. (2017b), each model uses L2-norm regularization of filter weights to encourage loudness invariance with the exception to the harmonic CNN-based models, which use a weight decay of $1e$–4 (Won et al., 2020).

6.2.1.1 Vertical filter network

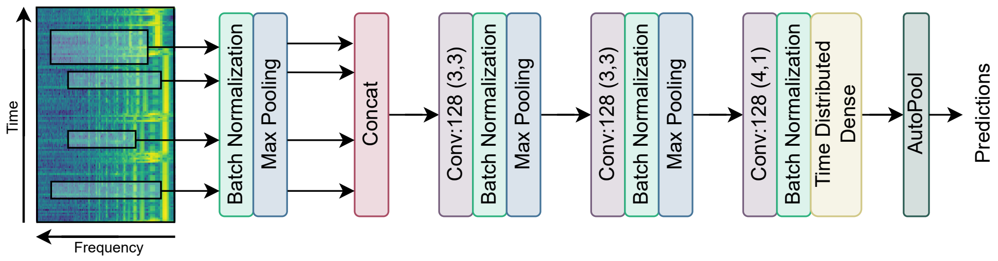


Figure 6.2: Block diagram showing the configuration of the vertical filter network with auto-pooling.

The vertical filter network (VF-CNN) is based on the *multi-layer* architecture in Pons et al. (2017b) for predominant musical instrument recognition. Figure 6.2 provides an overview of the VF-CNN configuration. The input spectrogram is set to be of size 500×128 to accommodate for longer observations of audio loops (see Section 6.2.2). The front-end utilizes several vertical convolution filter sizes (black rectangles in Figure 6.2) to efficiently model timbral characteristics present in the spectrogram. Custom filter sizes are used to capture both wide (e.g., bass, chords) and shallow spectral shapes (e.g, percussion). The numbers and sizes of filters used in the front-end are as follows: 128 filters of sizes 5×1 and 80×1; 64 filters of sizes 5×3 and 80×3; and 32 filters of sizes 5×5 and 80×5.

All convolutions in the front-end use *same* padding, and max-pooling is applied to obtain a 16×16 summary of each feature map. This is followed by two 2-D convolutional layers with batch normalization (Ioffe & Szegedy, 2015) and Exponential Linear Unit (ELU) (Clevert et al., 2016) activation functions. The first 2-D convolutional layer is followed by strided $(2, 2)$ max-pooling. After the final 2-D convolutional layer, we experiment with two pooling operations to summarize the information learned by previous layers prior to predictions (see Subsection 6.2.1.4).

6.2.1.2 Square filter network

The square filter network (SF-CNN) contains four 2-D convolutional layers with 128 small-rectangular filters of size 3×3 and *same* padding. After each convolutional layer, batch normalization is applied with an ELU (Clevert et al., 2016) activation function. Each convolutional layer is followed by strided $(2, 2)$ max-pooling, with the exception of the final convolutional layer, which also uses one of the two summarization pooling operations described in Section 6.2.1.4.

6.2.1.3 Harmonic CNN

In Ching et al. (2020), AIRC was approached using a CNN with a data-driven harmonic filter-based front-end (H-CNN) (Won et al., 2020). We re-implement this architecture and use it as a baseline to test our proposed models. The input t is passed through a set of triangular band-pass filters to obtain a tensor representing it as six harmonics. Harmonic structure is captured by treating the harmonics as channels and processed by a 2-D CNN. The CNN consists of seven convolution layers and a fully connected layer. All but the final convolutional layer is followed by 2×2 max-pooling, batch normalization and a Rectified Linear Units (ReLU) activation function. Global max-pooling is applied to the final convolutional layer. The output layer is a 5-way fully-connected layer with, a sigmoid activation function and a 50 % dropout.

6.2.1.4 Summarization Pooling

We consider two pooling operations for summarizing the information learned in the final convolutional layers: auto-pooling and standard global max-pooling.

Auto-pool is a trainable pooling operator capable of adapting to data characteristics by interpolating between min-, max-, or average-pooling (McFee et al., 2018). For the configurations that use auto-pooling, the final convolutional layer uses as kernel size (4,1). This is followed by batch normalization and a time-distributed dense layer with a sigmoid activation function and r output nodes, where r is equal to the number of classes. The output of the time-distributed dense layer is fed to the auto-pooling operation, which produces the final predictions.

For configurations that use GMP, the final convolutional layer is summarized with global max-pooling and then fed to a fully-connected output layer consisting of r output nodes, sigmoid activation functions and a 50% dropout.

6.2.1.5 Loss function

The loss function used for updating the parameters of each model is Binary Cross Entropy (BCE). BCE can be calculated as:

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\, p(y_i)) + (1-y_i)\cdot \log\,(1-p(y_i)) \tag{6.1}$$

where N represents the number of examples in the training set and $p(y_i)$ is the predicted probability of the i^{th} example.

6.2.2 Network Training

Input audio is pre-processed through resampling and conversion to a spectrogram representation. Audio loops are resampled to 16kHz and the Short-Time Fourier Transform (STFT) of each loop is calculated using a window size of 512 samples and a hop size of 256 samples. For the H-CNN, magnitudes of STFT are provided as input to the model. For the SF-CNN and VF-CNN, the inputs are log-scaled Mel spectrograms with 128 Mel-frequency bands.

All models are trained using the Adam optimizer (Kingma & Ba, 2015) with a learning rate $1e$–4, where each iteration takes a mini-batch of 8 examples. All weights are initialized using He's constant (He et al., 2015) to promote equalized learning. Early stopping was used to complete the training once the model performance ceases to improve over 15 epochs. The epoch that achieves the best accuracy on the validation set is used for testing.

6.2.3 Loop Activation Transcription

Loop activation transcription involves predicting the loop activations of instrumentation roles as they occur over time. Taking advantage of the grid-based structure and consequently fixed tempo of loop-based EM, we are able to use the proposed AIRC system to analyze the loop structure of a given composition. The AIRC system enforces separation between roles by design and does not rely on loops being an exact repetition of themselves, thus making it robust to variation such as automation and resequencing.

Instrumental role predictions for full EM compositions are obtained by passing an audio file into the AIRC system in 4-bar segments and assessing output activations. Segmenting a full-length EM composition into 4-bar loops, on which we then perform AIRC, instrumentation role activations may be derived for each loop, results in a form of EM transcription.

6.3 Evaluation

The AIRC model presented in Section 6.2 is assessed through two evaluations to determine: 1) AIRC performance using the various configurations and augmented version of the FSLD, and 2) performance for loop activation transcription.

6.3.1 Automatic Instrumentation Role Classification

6.3.1.1 Evaluation Methodology

The architectures (i.e., VF-CNN, SF-CNN, and H-CNN) and pooling strategies (i.e., auto-pooling and GMP) presented in Section 6.2 are evaluated in order to determine the optimal configuration for AIRC. Following Pons et al. (2017a), we use two sets of performance measurements: Area Under the Receiver Operating Characteristic Curve (ROC-AUC) and Area Under the Precision-Recall Curve (PR-AUC). The metrics were calculated on the test set for each of the models under evaluation.

In Ching et al. (2020), we also calculate the F1 score; however, we omit this evaluation metric as it depends on a decision threshold applied to the per-class output scores; whereas, ROC-AUC and PR-AUC measure model performance globally, integrating all possible thresholds.

6.3.1.2 Augmented Freesound Loop Dataset

To train and evaluate the different models we use the FSLD (Ramires et al., 2020c), comprising of various loops uploaded to Freesound (Font et al., 2013) under Creative Commons licensing. Of the various annotations present within the FSLD, we use tempo, key and loop instrumentation roles. The most important of which is the instrumentation role—a multi-label annotation for which the possible roles are: percussion, bass, chords, melody, sound Fx and vocals.

The original FSLD contains 2936 loops, of which 1531 have only one instrumentation role and 1405 have more than one. As can be seen from the class distribution in Table 6.1, the classes in this dataset are heavily imbalanced.

Percussion	54.95	Fx	24.80
Bass	19.10	Melody	21.31
Chords	11.90	Vocal	2.29

Table 6.1: Distribution (%) of instrumentation roles in FSLD.

In order to adapt this dataset to our task, we apply modifications to the data. We first remove all vocal loops as they do not provide sufficient training and testing material. All remaining loops are time-stretched to 120 Beats Per Minute (BPM). Longer loops are cropped to a length of 4 bars (i.e., 8 secs), while loops shorter than 4 bars are cropped to either 1 or 2 bars and repeated to a length of 4 bars. We separate loops which have multiple instrumentation roles from those which only have one, and randomly select 70% of each for training and 30% for validation and testing. From the latter split, 60% are used for testing and 40% for validation.

Besides using the previously described training set of the FSLD, we applied a data augmentation procedure to handle the main imbalance issues on the dataset. These are 1) the lesser presence of loops with more than one instrumentation role (i.e., multi-label) compared to the ones with just one role (i.e., single-label) and; 2) the number of loops for each instrumentation role class, shown in Table 6.2.

The data augmentation procedure utilizes common production techniques that are used in commercial music recordings including key matching, tempo matching and the use of audio Fx such as distortion, reverb and chorus.

Percussion	929	Fx	222
Bass	92	Melody	174
Chords	102		

Table 6.2: Distribution of the loops with only one instrumentation role in FSLD.

To balance the number of loops per class, we use an augmentation methodology similar to the one proposed in Chapter 4. The loops are processed through several effects, including delay, bitcrusher, chorus, flanger, reverb, tube saturation and pitch-shifting, resulting in 1000 loops for each of the r classes under observation ($r = 5$), totaling 5000 loops.

We create multi-role data by overlapping loops from each augmented single label class such that all single and combined classes contain the same number of loops. We start by calculating the possible combinations $\binom{r}{k}$, where k is the number of instrumentation roles in the combination ($2 \leq k \leq 5$). To balance the dataset in both the number of loops per instrumentation role and in k, the number of augmented loops (5000) is divided by $\binom{r}{k}$ to obtain the number of loops required for each combination (e.g., for k=2, $5000/\binom{r}{k} = 500$ for each combination of roles). The final loops are then created by harmonically combining the single instrumental role loops. We combine only loops with compatible modes (e.g., Major with Major), and pitch shift the selected loops to their average key. Discarding the original multi-label loops of the training set, this process results in a total of 25000 loops that can can used for training. In order to evaluate the effect of this augmentation procedure (Aug), we compare the accuracy of the models trained with those trained with the original dataset (FSLD) on the same test and validation data.

Percussion	27.59	Fx	23.17
Bass	20.33	Melody	18.15
Chords	10.77		

Table 6.3: Distribution (%) of instrumentation roles in the test set.

6.3.2 Loop Activation Transcription

6.3.2.1 Evaluation Methodology

To investigate the capacity of the AIRC system for transcribing loop activations in EM compositions, we compare all the AIRC configurations (Section 6.2). The best performing configurations are then compared with the results of the previous approach to loop activation transcription by Smith & Goto (2018).

As in López-Serrano et al. (2016) and Smith & Goto (2018), we evaluate the loop activation predictions against a ground truth in terms of accuracy. As accuracy expects a binarized transcription, we use a repeated k-fold cross validation together with a grid search to identify the best threshold for binarising the predictions of each role. In order to investigate the generalization of the proposed models, we use 2-fold cross validation repeated 10 times, where one fold is used as a validation set to identify thresholds and the other is reserved

Model	**Dataset**	**Pooling**	**Param.**	**PR-AUC**	**ROC-AUC**	**Bass**	**Fx**	**Perc.**	**Chords**	**Melody**
H-CNN	Aug	GMP	3619986	59.18	77.34	40.30	57.05	94.60	47.92	56.01
H-CNN	Pure	GMP	3619986	61.83	80.39	53.65	42.21	94.10	60.30	58.89
VF-CNN	Aug	GMP	1098869	65.60	80.99	47.11	64.92	97.62	63.11	55.22
VF-CNN	Pure	Auto	1102394	66.98	82.52	57.59	66.43	95.75	50.37	64.77
VF-CNN	Aug	Auto	1102394	67.47	81.40	46.18	67.10	97.05	56.13	**70.89**
SF-CNN	Aug	Auto	**313674**	68.15	82.19	55.12	68.30	**98.03**	58.81	60.49
SF-CNN	Aug	GMP	445701	68.40	81.59	**62.21**	59.80	95.93	62.39	61.68
SF-CNN	Pure	GMP	445701	68.74	83.83	58.72	63.11	95.97	64.74	61.14
VF-CNN	Pure	GMP	1098869	70.62	**85.72**	53.83	**71.73**	97.84	64.90	64.78
SF-CNN	Pure	Auto	**313674**	**71.28**	85.12	57.76	59.18	95.98	**73.20**	70.30

Table 6.4: AIRC performance (%) and model size for each configuration, where bold indicates highest scores.

for computing accuracy against the ground truth. Thresholds for each class are identified by performing a grid search over a range between 0.01 and 1 with a step size of 0.01, then selecting the thresholds which provide highest accuracy on the validation set.

In Smith & Goto (2018), approaches which require the downbeat tracking are considered *guided*. As our proposed approach requires BPM annotations for time-stretching, we only compare our models with the *guided* algorithms.

6.3.2.2 Dataset

We apply our proposed models to the dataset used in López-Serrano et al. (2016) and Smith & Goto (2018). The dataset consists of simplified EM compositions built by generating templates similar to the ones in Figure 6.1 with 4-bar loops. We refer to this as the *Artificial* dataset for the reason that the the loops are repeated without variation, which would usually be achieved in professional music through DAW techniques, such as automation and resequencing.

The automatic arrangement method provided in Smith & Goto (2018) is used to build 21 music compositions with seven genres and three templates–*composed*, *factorial* and *shuffled factorial*. For the *composed* template, loops are introduced and removed in an iterative manner. The *factorial* template contains all possible combinations of loops, arranged iteratively. The *shuffled factorial* template contains the same loop combinations, with shuffled ordering. *Factorial* and *shuffled factorial* datasets are useful for seeing how the models perform on all of the loop combination possibilities for the *Artificial* dataset, whereas *composed* layout is more representative of typical EM compositions in regards to the way that loops are iteratively introduced and removed throughout the composition.

Following the AIRC procedure, compositions are time-stretched from their annotated tempo to 120BPM and divided into 4-bar loops, which are provided as input to the AIRC systems.

6.4 Results & Discussion

6.4.1 Automatic Instrumentation Role Classification

The models are evaluated using use the Macro Average (MA) of the PR-AUC and of the ROC-AUC as a global metric. For individual instrumentation roles, we only show the PR-AUC. Due to the imbalance of the FSLD, which also affects the test set (Table 6.3), MA is used to provide an average accuracy over each class.

Table 6.4 presents the results of our AIRC experiment for the models discussed in Section 6.2, in which each model is presented in ascending order of their average PR-AUC. The ROC-AUC performance measure is consistently higher than PR-AUC; however, this metric can lead to over-optimistic scores when the dataset is unbalanced (Davis & Goadrich, 2006).

The best performing models *w.r.t* PR-AUC, are the SF-CNN with auto-pooling (71.28%) followed by the VF-CNN with GMP (70.61%). Both models surpass the current state-of-the-art, H-CNN trained on FSLD (61.82%) by a substantial margin. The SF-CNN mostly performs better than its VF-CNN counterpart. Vertical filters have been demonstrated to produce comparatively better results with tonal musical audio (Pons, 2019); however, the results of our evaluations suggest that square filters generalise better to the non-standard types of audio associated with EM.

The overall best performing model in terms of PR-AUC is the SF-CNN with auto-pooling trained on the Pure dataset. However, by closely inspecting the results achieved for individual instrumentation roles, it can be seen that it surpasses by almost 10% the PR-AUC achieved by other models in the *Chords* class, while not achieving such a high result in *Bass*, *Fx* and *Percussion*.

The highest performing instrumentation role for all models is *Percussion*, which was expected due to this role having the largest number of examples in the FSLD dataset. The roles that generally perform worst are *Bass* and *Chords*, which have the smallest number of examples in the FSLD. The performance of *Bass* has a considerable increase when using a combination of the SF-CNN with GMP and augmented data. Additionally, *Chords* performs significantly better when using the SF-CNN and auto-pooling configuration trained with the Pure dataset.

The best three performing models in terms of PR-AUC are trained on the Pure dataset, followed by the Augmented one. However, it can be seen that the *Bass*, *Percussion* and *Melody* roles tend to benefit from training with the Augmented dataset. As the configurations perform better for different classes, it is possible to use a combination of the models for classifying individual instrumentation roles. This combination would lead to an average PR-AUC of 75,213%, substantially surpassing each model.

6.4.2 Loop Activation Transcription

Table 6.5 presents the loop activation transcription results using the AIRC configurations (Section 6.2) to transcribe the compositions in the *Artifical* dataset. Each model is presented in ascending order of their mean classification accuracy over the instrumentation roles. Additionally, Table 6.5 provides the classification accuracy for each individual role (*Bass*, *Drums*, *Fx* and *Melody*).

Model	Data	Pooling	Mean	Bass	Drums	Fx	Melody
H-CNN	Pure	GMP	75.1	71.8	96.1	55.6	76.7
H-CNN	Aug	GMP	79.5	53.1	95.8	81.6	87.6
VF-CNN	Pure	Auto	80.2	63.7	98.6	63.1	**95.3**
VF-CNN	Pure	GMP	80.9	69.0	99.3	65.8	89.4
SF-CNN	Pure	Auto	81.0	66.9	97.3	71.6	88.4
SF-CNN	Pure	GMP	81.8	69.2	**100.0**	63.4	94.6
VF-CNN	Aug	Auto	82.5	**74.2**	99.7	79.7	76.6
VF-CNN	Aug	GMP	84.7	71.7	**100.0**	75.7	91.4
SF-CNN	Aug	GMP	86.2	71.7	**100.0**	79.6	93.2
SF-CNN	Aug	Auto	**86.9**	68.3	**100.0**	**85.7**	93.4

Table 6.5: Loop activation transcription accuracy (%) results for AIRC configurations, where bold indicates highest scores.

The overall best performing model uses the SF-CNN with auto-pooling configuration trained using the augmented dataset (86.9%) followed by the SF-CNN with GMP (86.2%). For this task, models trained with the augmented dataset generally appear to outperform those trained with Pure dataset, which could be due to the fact that the augmentation process ensures there is a balanced distribution of all possible role combinations and it is common in the compositions for several roles to be active in a single loop. Drums are classified most accurately for all model configurations with four models achieving 100% accuracy. This is expected as *percussion* has the largest number of samples in the FSLD dataset, and is usually the most prominent element in EM compositions. In some cases, the VF-CNN configuration seems to improve performance of *Melody* and *Bass* roles, which could suggest that the classification of roles containing melodic instruments benefit from using vertical filters at the front end of the system.

Figure 6.3 presents loop activation transcription results for the three template variations using our two best performing AIRC configurations (i.e., SF-CNN-AUTO and SF-CNN-GMP) compared with the NTF (Smith & Goto, 2018) and NMFD (López-Serrano et al., 2016) methods previously proposed for this task.

On a glance, we can see our architectures out perform the previous methods in regards to accuracy for the *composed* layout, with SF-CNN-GMP (red) achieving the highest score. NTF (blue) achieves the best performance for the *factorial* layouts closely followed by our SF-CNN-AUTO architecture. Furthermore, the AIRC system has a considerably faster runtime than NTF (∼30 secs per composition) and NMFD (∼10 mins per composition). Predictions for a full EM composition are calculated in under a second using AIRC, which could be beneficial when analyzing large collections of music in DJ software. As mentioned in Smith & Goto (2018), an additional shortcoming of the NTF and NMFD approaches is that the algorithms depend on loop roles not co-occurring throughout the composition. The proposed AIRC approach enforces independence between the different roles, thus making it more suitable for transcribing loop activations of real-world EM compositions, in which loops often vary through automation and resequencing.

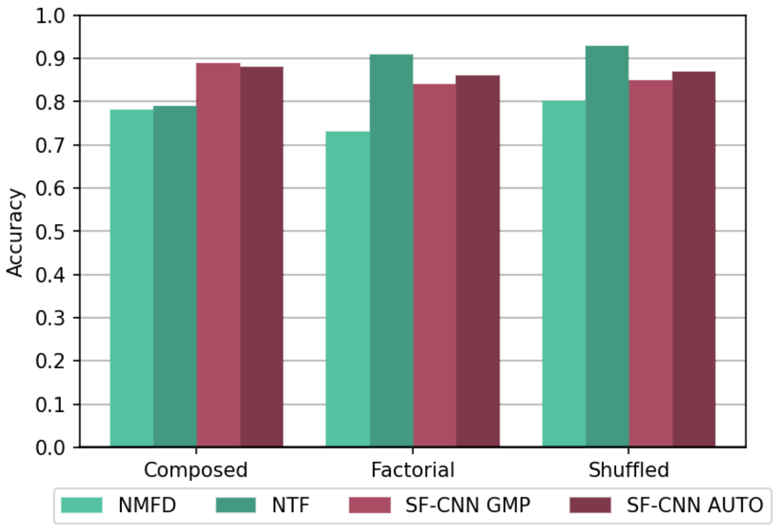


Figure 6.3: Loop activation transcription accuracy scores.

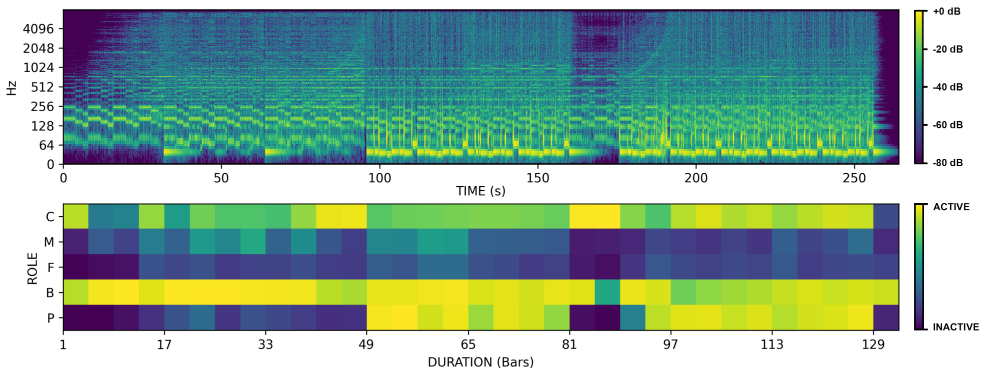


Figure 6.4: Estimated loop activation structure of *Joyspark* (2020) by Om Unit using our proposed model. Log-scaled STFT power spectrogram of the EM composition (*top*) and estimated templates corresponding to the loop activations showing predictions for each class: Chords (C), Melody (M), Sound Fx (F), Bass (B), and Percussion (P) at 4-bar intervals (*bottom*).

6.4.2.1 Real World Scenarios

Our approach to loop activation transcription with AIRC can be applied to full-length, professionally produced EM, which has not been explored in previous literature.

An Instrumentation Role Activation Map (IRAM) of the EM composition *Joyspark* (2020) by Om Unit[54] using the proposed method for loop-based EM structure analysis (Section 6.3.2) is presented in Figure 6.4. For visualization and comparison, we show a log-scaled STFT power spectrum of the EM composition above the IRAM. The IRAM allows us to visualize activations for each role over the duration of the EM composition, where each square is a measurement of four bars. Furthermore, we can see how each role develops throughout the EM composition. For example, the melody role activations progressively increase between bars 1–41, which corresponds with a synthesizer arpeggio that is gradually introduced by automating the cut-off frequency of a low-pass filter. Additionally, the chord role activations increase between bars 1–49 in correlation with the chords in this section that gradually increase in volume. Activations for the percussion role also correlate well with the composition as can be seen between bars 49–81 and 97–129—the only sections that contain percussion. Finally, the key structural sections of the composition are easily identifiable. For example, the introduction to the composition (bars 1–49) begins relatively sparse in the composition and IRAM; whereas, bars 49–81 and 97–129 are quite clearly the *core* of the piece—that is, the most energetic sections of the composition typically established by the *drop* (Yadati et al., 2014).

Additionally, the transcription enabled by our system could help EM producers identify sections of music that contain specific roles. For example, this would be useful for finding breakbeats (i.e., percussion-only passages) in digital music recordings (López-Serrano et al., 2017).

6.5 Conclusions

In this chapter, we have introduced the task of Automatic Instrumentation Role Classification (AIRC) of loops. We propose a method that utilizes a novel data augmentation method and CNN-based architecture with auto-pooling. The evaluation results show that we achieve a high accuracy score, allowing for a more reliable transcription of loops in unstructured collections of audio. Furthermore, we have introduced a deep learning approach for estimating the structure of loop-based electronic music and compared it with previous loop

[54] https://omunit.bandcamp.com

activation detection methods. Our approach achieves comparable results while achieving a considerably faster computation time.

The IRAM derived from our system has many potential use cases in music production and performance. MIR tasks that rely on structural information could benefit from this transcription (e.g., automatic DJing (Vande Veire & De Bie, 2018), music mashups (Davies et al., 2013)). The IRAM could be used as a visual aid for DJs to anticipate upcoming sounds (e.g., drums, bass) or to help to identify key structural events in EM (Yadati et al., 2014).

A possible direction for future research in this area would be to train the system using a smaller timescale (e.g., 1-bar measures) to achieve higher resolution transcription of instrumentation role activations. Additionally, as no annotations for ground-truth instrumentation roles exist for real-world EM compositions, future work could involve annotating a corpus of these recordings for the evaluation of this task.

Part II

Automatic Generation of Music Loops and Instrument Samples for Electronic Music Production

CHAPTER 7

Introduction and Overview of Creating Percussive Sounds

In this part of the dissertation, we focus our attention on the task of generating drum sounds using deep learning in a controllable manner. Browsing and generating sounds are two different music creation tasks which share the same goal: finding the ideal sound for a composition. The sounds retrieved through browsing are limited to the sound collection being searched, while sound synthesis is limited by the sonic characteristics of the sound source. Using generative deep learning models trained with large sound collections, we can develop a synthesizer which is able to create sounds with a similar diversity as the collection. Generative models, besides being able to generate sounds similar to the ones seen during training, can create the *in-between* of these sounds and offer new sonic possibilities. The generation of pitched sounds using deep learning has seen a significant amount of research, while the synthesis of unpitched percussive sounds has been largely ignored. On this part of this dissertation we will therefore investigate the generation of percussive sounds using generative deep learning models in a controllable manner.

7.1 Percussion and Cultural Context

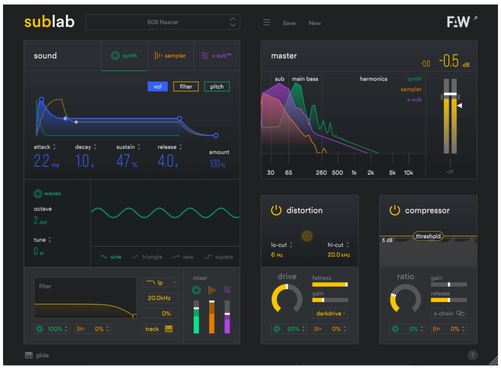
Percussion is one of the main components in music and is normally responsible for a song's rhythm section. Percussion instruments have been part of human life for a long time. Drums which use alligator skins as the membrane were discovered in China, dating back to the Neolithic period (Liu, 2005). Since then, percussion has accompanied human evolution in a variety of situations: they have been used in ceremonies, to support rituals and even for military purposes. In music, percussion has been a core part of the music made in the last centuries. It is present in classical music orchestras, in rock and jazz bands and, more recently, in Electronic Music (EM). In this genre, percussion and rhythm are one of the things that define sub-genres, where each is commonly associated with a set of drum timbres and rhythm patterns. To mention some examples, Dubstep typically uses a bright snare with a long decay, House is focused on a round kick drum with a crispy snare, Classic Hip Hop employs gritty acoustic drums obtained from Funk records, while Trap relies on electronic long kick drums with synthesized hi-hats. The drum sounds are therefore responsible for a big part of the feel of EM pieces. Choosing the perfect drum sound in an efficient manner is extremely important, can help speed up the whole composition process and even inspire the creator.

7.2 Creating Percussive Sounds

Most common percussion instruments create sound when struck or scraped, while others emit sound when plucked, rubbed or even shaken (Latham, 2011). Traditionally, timbral characteristics of percussion were obtained by modifying the acoustic properties of the instruments themselves – for instance, the use of different shell configurations or materials for constructing drums, or different shapes and alloys for cymbals. Developments in analogue synthesis paved the way for creating and designing percussive sounds electronically. New electronic instruments were developed for generating these sounds either through playing prerecorded samples or through synthesizing them. These are called drum machines and became very popular for EM (Hasnain, 2017). As an example, drum machines such as the Roland TR-808 generated sounds by combining synthesized tones with white noise. These drum machines became a staple in a variety of music genres including Hip Hop, House and Techno. However, these early drum machines did not provide much control over the generation of the sounds. With the developments in digital audio technology and computer music, new drum machines were hand-designed using expert knowledge of synthesis techniques and Electronic Music Production (EMP). More recently, with the development of Digital Audio Workstation (DAW) and other music-making software, digital drum sound creation became common practice for music makers of all backgrounds, enabling advanced digital signal processing techniques to be applied to drum sound design. Typically, a music maker either selects drum sounds from their sound collection or creates it using a dedicated software. These programs normally display parameters related to the synthesis process such as the pitch, the energy envelope or effects applied on top and can have presets (curated sounds where the parameters are selected by an expert sound designer) to provide a starting point. In Figure 7.1, we present interfaces for two well known drum synthesis software: FAW SubLab[55]

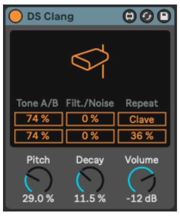
[55] https://futureaudioworkshop.com/sublab

and the Ableton Drum Synthesis (DS) devices. FAW SubLab provides a high-level of control, perfect for expert users who know exactly what each control can do. On the other hand, Ableton DS devices provide simple controls for creating the percussion sounds, which is appropriate for novices.



attack 2.2 ms, decay 1.0 s, sustain 47 %, release 4.0 s, amount 100 %, master -0.5 dB

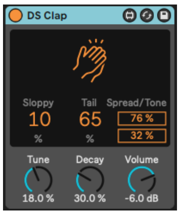
(a) FAW SubLab


DS Clang

Tone A/B	Filt./Noise	Repeat
74 %	0 %	Clave
74 %	0 %	36 %

Pitch	Decay	Volume
29.0 %	11.5 %	-12 dB

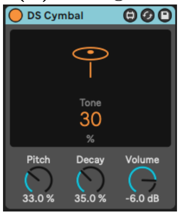
(b) Clang DS


DS Clap

Sloppy	Tail	Spread/Tone
10 %	65 %	76 %
		32 %

Tune	Decay	Volume
18.0 %	30.0 %	-6.0 dB

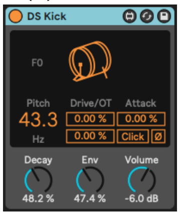
(c) Clap DS


DS Cymbal

Tone 30 %

Pitch	Decay	Volume
33.0 %	35.0 %	-6.0 dB

(d) Cymbal DS


DS Kick

F0 Pitch	Drive/OT	Attack
43.3 Hz	0.00 %	0.00 %
	0.00 %	Click

Decay	Env	Volume
48.2 %	47.4 %	-6.0 dB

(e) Kick DS

Figure 7.1: Example of interfaces for drum synthesis software. On the left FAW Sublab and on the right the drum synthesizer modules from Ableton Live 11.

Concurrently, aesthetics have have evolved up to a point where virtually any sound can be used as a percussive sound as long as it has some energy envelope characteristics. Timbre does not need to be related to the one of traditional percussion instruments or sound machines anymore, and creativity in percussion sound design is strongly encouraged in EM.

7.3 Music Creation Using Deep Learning

Recent advances in deep learning introduced novel methodologies for synthesizing data. Instead of relying on experts for designing systems to generate specific kinds of data, these methodologies are data-driven: the algorithms learn how to represent the distribution of data on which they are trained. Learning this representation allows the creation of new examples similar to the data it saw during training. In the case of digital sound, a generative model can be taught how to create new audio waveforms (sample by sample), spectograms or other representations of sounds. The controls provided by these models are typically very different from the ones provided in a synthesizer. While synthesizers commonly employ energy envelope, additive and subtractive synthesis for generating sound, generative models create sound from points in a high-dimensional space called the latent space. This representation is optimized to encode the audios it was trained with and enables the generation of similar examples, without needing knowledge about sound synthesis. Architectures such as Autoregressive Networks (Oord et al., 2016a; Engel et al., 2017; Mehri et al., 2017), Variational Autoencoder (VAE) (Kingma & Welling, 2014), and Generative Adversarial Network (GAN) (Goodfellow et al., 2014) have all been proven to generate high-quality results in a variety of domains, from images of human faces to musical audio, as will be further explained in Chapter 8. In the image domain, these also enable new techniques for creating data: Style transfer allows to compose one image on the style of another image, interpolation permits to generate what is *in-between* two images by navigating a latent space and image-to-image translation can create highly realistic images from drawings.

In parallel, recent advancements in generative deep learning based methodologies have sparked a huge interest in music generation. Research has focused on both the symbolic and the audio domains. On the symbolic side Dong et al. (2018) use GANs to generate music pieces, Brunner et al. (2018) and Cífka (2021) have experimented with transferring the style of music pieces, while Grachten et al. (2020) and Lattner & Grachten (2019) are able to generate bass and drums based on the rest of the music piece. Interesting possibilities for music making using deep learning methodologies in the audio domain include timbre transfer, music source separation and generating musical sounds such as singing voice and instrumental notes. Timbre transfer was presented by Engel et al. (2020) and permits transforming a recording into a different instrument directly on the audio domain. Music source separation allows the decomposition of music recordings into their different elements such as bass, drums, melody and harmony. This can enable new methods of sampling parts of music pieces without the need for each element to appear alone. Source separation has been a deeply studied topic with some example literature presented by Défossez (2021) and Stöter et al. (2019).

Our aim is to explore generative deep learning methodologies for making the task of creating and navigating percussive sounds easier. We target conditional synthesis based on high-level semantic parameters and automatically discovering these parameters from unconditional models. The models we present can be trained with private user collections, enabling the generation of samples similar to the ones present in the collection and, through interpolation, generating the *in-between*.

In **Chapter 7** (this chapter), we started by presenting how the sound of drums have been created throughout human history, from its inception to the possibilities of drum synthesis offered by modern software. We also briefly described some of the possibilities that deep learning presents towards new synthesis techniques, which will be further detailed in **Chapter 8**. We will then explain our contributions towards generating both percussive one-shots (Chapter 9) and loops (Chapter 10) in a controllable manner. We will wrap up with Chapter 11 where we compare different control interfaces for generative models.

CHAPTER 8

Literature Review on Sound Generation

In this chapter, we provide an overview of the approaches proposed for generating musical audio using deep learning methodologies. We will present the fundamentals of generative modeling using deep learning techniques and the main architectures which are used. We will also present interesting applications from other domains which use these set of algorithms. We then provide an overview of techniques applied to provide some user control to the generative process. Finally, we do an in-depth review of how generative algorithms have been used for music creation, with a strong focus on drum synthesis.

8.1 Deep Generative Models

We can define generative models as any model that, given a training set of examples of a distribution X, somehow learns to represent an estimate of it (Goodfellow, 2017). Deep generative models use deep learning to address this task and have shown great progress in both generation quality and time. Different deep learning architectures have been proposed and applied to several tasks: from generating new examples similar to the training data, to dimensionality reduction and feature learning, data denoising, outlier detection and, more recently, creating or assisting the creation of art.

The task of generating data that fits a distribution is significantly different from the classification tasks studied in Chapter 3, especially when done in an unsupervised manner. Instead of finding a function which maps data x to labels y, in unsupervised learning the main goal is to learn an underlying structure of the training data. Discriminative models, learn a probability distribution $p(y \mid x)$, while generative models learn the probability distribution $p(x)$. One common method to learn this density function is the principle of maximum likelihood: find the parameters for a model that maximize the likelihood of the training data. A deeper discussion on the optimization of maximum likelihood can be found in Goodfellow (2017).

Some generative models can assign density functions to the training data (explicit), some can only generate samples from the learned distribution (implicit), and some can do both. Models with explicit density functions allow knowing the likelihood of a new data point fed into the input. On the other hand, models with implicit density functions (e.g. GANs based on likelihood) do not allow extracting a likelihood value, but allow sampling from the underlying distribution. The explicit density functions can be either tractable (in Autoregressive models) or approximate (Variational Autoencoder (VAE)). Tractable density functions output the actual value for the density function while models which compute approximate density functions can only compute an approximation to this probability.

In the following subsections, we will explore the most commonly used generative model architectures, their inner workings, example applications, advantages and disadvantages.

8.1.1 Autoregressive Models

Autoregressive models treat each data example x as a sequence of points x_i. If we assume that x has T elements, $p(x)$ can be decomposed as in Equation 8.1. The chain rule from probability theory exemplified in Equation 8.2 allows to break down $p(x)$ into the probabilities of each point given the previous points.

$$p(x) = p(x_1, x_2, x_3, ..., x_T) \tag{8.1}$$

$$p(x_1, x_2, x_3, ..., x_t) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)... \tag{8.2}$$

Autoregressive networks explicitly model the correlations between the different points x_i in a sequence x. Therefore, they can output a probability for a full sequence x as well as to each point. Each generated point is conditioned on the previous points and, consequently, only one point can be generated at a time. Hence, autoregressive networks are very good at modeling the small-scale dependencies of the data.

One of the main families of autoregressive models are the Recurrent Neural Networks (RNNs), which use hidden states h_t to store information of the previous points of the sequence. This hidden state is calculated according to the following equation: $h_t = f_W(h_{t-1}, x_t)$, where x_t is the current segment, h_{t-1} the previous hidden state and f_W is the learnable network function with weights W. In a vanilla RNN, $f_W(h_{t-1}, x_t) = tanh(W_{hh}h_{t-1} + W_{xh}x_t + V_h)$, where W_{hh} and

W_{xh} are the learnable matrices for the hidden state and the input, and V_h the learnable bias. For the output layer, RNNs have other trainable weights and biases W_hy and V_y, and their output can be calculated as $y_t = W_{hy}h_t$. The training procedure updates the weights W and the biases V, by backpropagating a loss function based on the timesteps of the generated example. An example of a RNN architecture is presented on Figure 8.1 A detailed description of RNNs and their training can be found at Sherstinsky (2020).

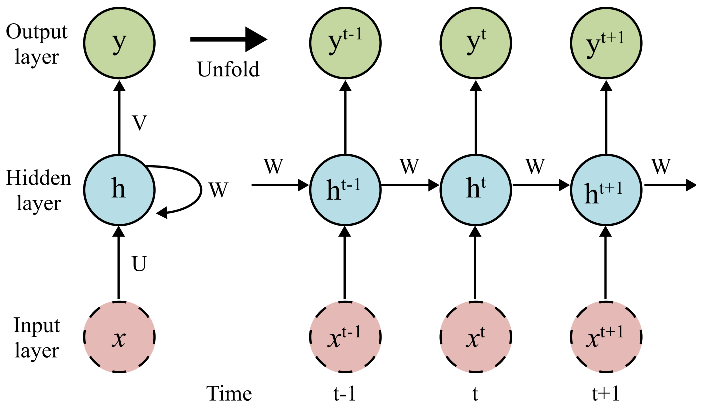


Figure 8.1: Schematic of an RNN architecture.

RNNs main issues are the long time they take to generate a complete sequence, as well as the difficulty in modeling large-scale dependencies of the data (long time dependencies in the case of audio). The long generation time is due to the networks only being able to generate one point at a time because of the explicit conditioning on previous points.

In an attempt to mitigate the long-term forgetting of RNNs, Wavenet has been proposed in Oord et al. (2016a). Instead of using RNN blocks, the prediction of the next element in a sequence is done using a CNN which uses casual dilated convolutions (Oord et al., 2016b). The causal aspect of the convolutions enforces that the model respects the order of the elements in the sequence and that it cannot depend on any of the future timesteps of the sequence. A dilated convolution is a convolution where the filter is applied to a longer sequence, by skipping input values according to a defined step. This enables the network to have a bigger receptive field and be able to understand larger timescales of the sequences it is trained with. Stacking more layers of dilated convolutions further increases the receptive field of the model. As the Wavenet does not use recurrent connections (as RNNs use the information of the previous state), the predictions for the different timesteps can be computed in parallel. Therefore, Wavenet is able to have a reduced training time, especially for long sequences. When generating new data, however, each new sample to be generated requires the previous point in the sequence, making inference fairly slow.

8.1.2 Autoencoders

Autoencoders (AEs) (Hinton & Salakhutdinov, 2006) are neural networks which aim to reconstruct an input signal, while learning efficient representations with a lower dimensionality than the input. This architecture can be divided into two components: the *encoder* and the *decoder*. The encoder e takes an example x of the training data as input and maps it to a lower dimensional representation $z = e(x)$. The decoder d then transforms the embedding z to a reconstruction of the training data $x' = d(e(x))$. The reconstruction error is backpropagated for the network to learn on how to improve. The use of a latent space z with a smaller dimension than x forces the AE to learn the most representative characteristics of the training data. AEs have shown significant performance on tasks such as dimensionality reduction and unsupervised feature learning.

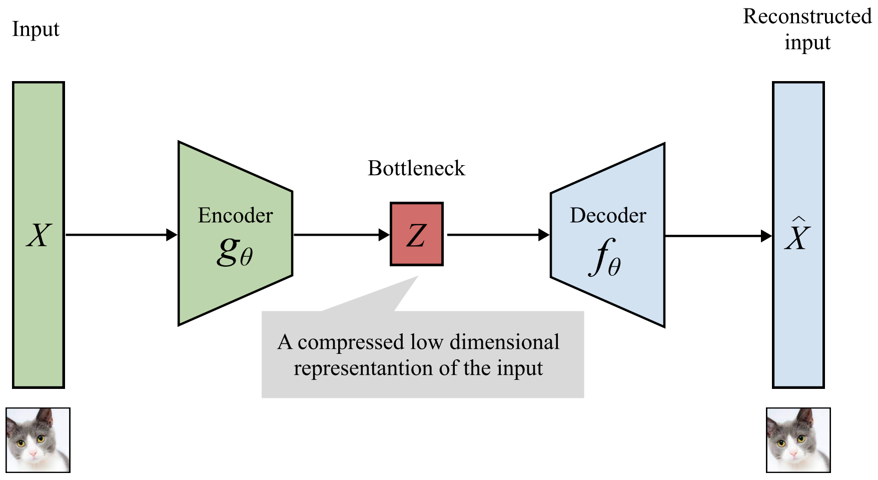


Figure 8.2: Autoencoder architecture.

One naive approach to generate new data using an AE would be to take a random point z from the latent space and regenerate it through the decoder $d(z)$. As there is no explicit regularization applied when creating the latent space, its continuity is not guaranteed and parts of it might not even be used by the network. Furthermore, AEs are not probabilistic models and, therefore, there is no reason for the distribution of the data generated from the latent space to follow the distribution of the training data. To tackle these issues, Variational Autoencoders (VAEs) (Kingma & Welling, 2014) apply a latent space regularization during training that avoids overfitting and ensures continuity (points close in the latent space should generate similar results when decoded) and completeness (any point from the latent space should decode to a realistic output).

Instead of encoding an input x as a single point, VAEs encode it as a distribution (typically a multivariate Gaussian) over the latent space. The training methodology is similar to AEs, after this encoding step, i) a point z is sampled from the encoded distribution; ii) z is reconstructed through the decoder $d(z)$, and iii) the reconstruction error together with a regularization penalty are calculated and backpropagated through the network. The regularization penalty is the Kulback-Leibler divergence between the encoded distribution and a standard Gaussian, which quantifies the similarity between these two distributions. Forcing the mean to be close to 0 enforces the latent space to encode similar examples together and to follow a similar probability distribution of the training data (the most common examples in the data will be in the center of the Gaussian latent space). Forcing the Gaussian's co-variance matrix to be close to the identity enables the continuity of the latent space by encouraging the encoded distributions to overlap.

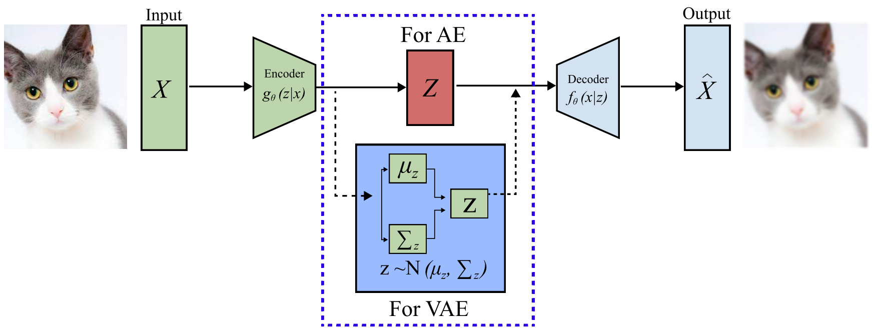


Figure 8.3: Difference between Autoencoders and Variational Autoencoders.

In order to generate new data from a VAE, we simply need to sample a point z from the multivariate Gaussian and decode it through $d(z)$. As the VAE learns the distribution of the trained data, this Gaussian sampling also forces the generated data to follow this distribution. Furthermore, we can use a trained VAE to edit data by encoding it to a latent distribution, sampling from it, and then modify the value of indexes of the latent point z.

Despite their fast training, generation and encoding time, generated samples from VAEs tend to be blurry, probably due to the assumed factorized distribution for the density function. To address this, this architecture has been combined with an Autoregressive decoder (Razavi et al., 2019), at the cost of longer generation time and possible training collapse.

8.1.3 Generative Adversarial Networks

The architectures which have been performing the best in terms of synthesis quality are the Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Karras et al., 2020). Although GANs can be made to work using the principle of maximum likelihood Grover et al. (2018), these normally employ a different type of learning paradigm: adversarial training. GANs are composed of two networks which compete against each other. One is the generator network, which aims at creating examples similar to the ones present in the training data by learning how to fool the discriminator. The discriminator network's job is to examine the examples created by the generator, as well as from the training data, and to distinguish which ones are real (from the training data) and false (from the generated data). The discriminator is a typical classifier network, which uses supervised learning to distinguish the real and false classes. On the other end, the generator is a network similar to an AE's decoder, which takes a random point z from a random distribution, typically Gaussian, and generates an example x' from it.

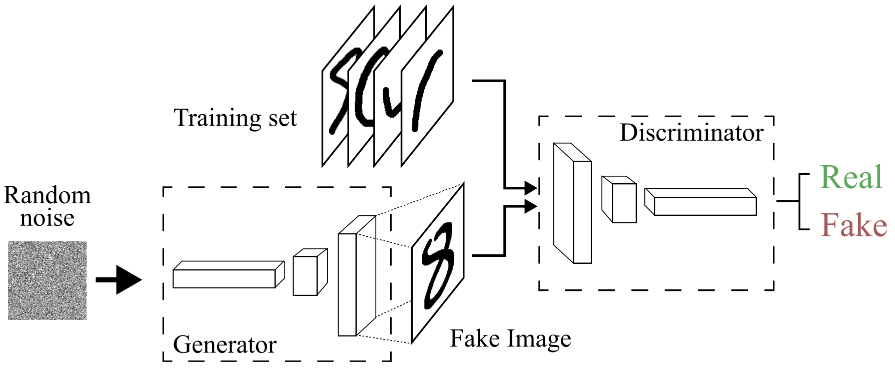


Figure 8.4: Generative Adversarial Network architecture.

This novel training methodology where two networks are learning in an adversarial manner at the same time comes with some possible issues. One is called Vanishing Gradients (Arjovsky & Bottou, 2017) and it occurs when the discriminator achieves optimal performance, not allowing the generator to learn anything more. Possible solutions to this include using the Wasserstein loss (Arjovsky et al., 2017) or modified loss functions. Another common issue with GANs is Mode Collapse, which happens when the generator creates very similar outputs with no variability that the discriminator finds plausible to be true. The network gets trapped in local minima and stops learning. As GANs normally do not use maximum-likelihood loss functions, they are prone to have a mode-seeking behavior, where just a part of the training data is represented.

Despite these disadvantages of GANs, these architectures have been showing the best synthesis quality in several domains, from image to audio. Several flavours of GAN with specific training methodologies and different goals have been proposed. To generate images from a distribution, the ProGAN (Karras et al., 2018) and the StyleGAN (Karras et al., 2020) enforce the networks to learn the different resolutions of the data, achieving impressing quality on the created images. GANs have also been employed to a variety of more creative tasks, from editing the attributes of a face given as input (Shen & Zhou, 2021), to transferring the "style" of an image to another image (Zhu et al., 2017), to conditional image generation (Isola et al., 2017).

8.2 Controlling Generative Models

In the previous section, we looked at some of the most used deep generative architectures. In order to be able to use them for creative purposes, we need to somehow control the generation process. Several approaches have been proposed to allow a degree of controllability, such as data curation, analysis and regularization of the latent space z, and conditioning of the network during training.

The most significant way to control what a network generates is by curating the data used for training. A network trained on a specific distribution of data is not likely to be able to generate data from a very different distribution. Therefore, the first step before training a model is to make sure that what we want to generate is well represented in the training data.

Once a model is trained, new data can be generated by sampling random points from the networks latent space. In an artistic scenario, the user most likely wants a deeper control of the generated data than random sampling. Projecting a data point x to the latent space (Creswell & Bharath, 2019; Zhu et al., 2016) allows the network to generate something similar to x and retrieve the embedding z for it, which allows to obtain a starting point for navigating the latent space.

Once we have a starting point z, we probably want to edit some characteristics of the generated data. To do this, we can manipulate the latent dimensions of z until we achieve the final result. Each dimension of the latent space should allow the manipulation of an individual or a set of features which capture the variability of the data. Recent work has shown that GANs can capture high-level semantic concepts (Yang et al., 2021) in the latent space dimensions. Finding what each dimension controls can be cumbersome and, therefore, several approaches have been proposed that allow the identification of the dimensions that control specific characteristics of the generated data. Supervised approaches (Yang et al., 2021; Jahanian et al., 2020; Shen et al., 2020) rely on the annotation or automatic classification of the characteristics of randomly sampled generated data from the model. On the other hand, dimensionality reduction techniques such as Semantic Factorization (SeFa) (Shen & Zhou, 2021) and GANSpace (Härkönen et al., 2020) employ Closed-Form Factorization or PCA to discover the directions in the latent dimensions that affect the most on the generated output.

Sometimes, however, the features the models learn while training do not relate to common human perceivable concepts or do not provide satisfactory control over the generation process. Having each dimension represent a different concept, i.e. a disentangled representation, provides a greater degree of control over the generation process. In order to explicitly enforce what the dimensions represent, two methodologies can be applied during the training process: latent space regularization and conditioning of the network. regularization techniques apply a new element to the loss functions that enforce the latent space to have certain characteristics. Chen et al. (2016) employ a latent loss that maximizes the mutual information between latent variables and the generated data. This results in a disentangled representation which allows the control of rotation and the width of digits when trained on the MNIST dataset (Lecun et al., 1998) and the presence or absence of glasses, hairstyles and emotion when trained on the CelebA dataset (Liu et al., 2015). In the audio domain, several regularization techniques have been applied to the generation of musical audio using VAEs. Esling & Bitton (2018) propose three regularization — an Additive Penalty Regularization, a Euclidean Regularization and a Prior Regularization — that aim to enforce the VAE's latent space to exhibit the same topology as several perceptually motivated timbre spaces.

Conditioning is a fairly different approach from the previously described ones and involves a new training paradigm. If we have extra information about our data, we can use it as a conditioning signal c and train the models to learn the conditional density distribution $p(x|c)$ instead of the distribution $p(x)$ we were trying to learn previously. The chosen conditioning signal c can vary in terms of how much information it contains, from low information signals such as class labels (e.g. if we want to generate the sound of a guitar or a piano) to very rich conditioning signals like one music piece we want to extract the drums from. Conditional neural networks enabled a variety of new tasks in the image domain, from generating digits from the MNIST dataset based on their classes (Mirza & Osindero, 2014), to generating realistic images from drawings or shapes (shown in Figure 8.5) (Isola et al., 2017) and even colorizing old black and white pictures (Zhao et al., 2021). Conditional models therefore provide a significant degree of control on the generation process, while maintaining great synthesis quality.

8.3 Generative Models for Assisting Music Creation

Audio synthesis technologies for music have been researched for many years, ranging from synthesizers generating pitched waveforms, to singing voice syn-

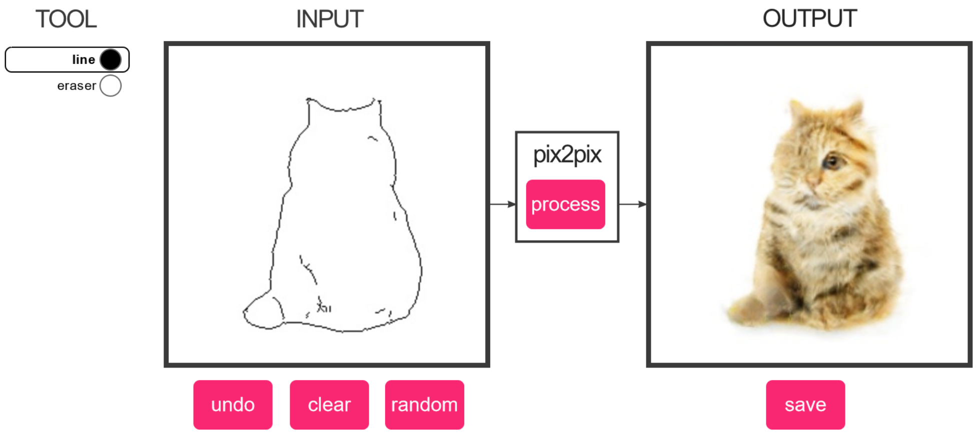


Figure 8.5: User interface for pix2pix conditional image generation

thesizers conditioned on melody and text[56], to deep learning based models capable of generating entire songs (Carr & Zukowski, 2018; Zukowski & Carr, 2018). In the context of assisting music creation, generative models have shown success especially in creating pitched instrumental sounds, when conditioned on musical notes. A pioneering work on this field was NSynth (Engel et al., 2017), a synthesizer based on the Wavenet autoregressive architecture, which, while capable of generating high-quality sounds, is very resource intensive. While the output of NSynth is subjectively similar to natural-sounding samples, the sequential nature of the model means that the processing time for generation is quite high, unless high-resource processing units are available. Wave-U-Net (Stoller et al., 2018), a feed-forward adaptation of the Wavenet architecture via probability density distillation has been proposed, showing that it is possible to directly map the input conditioning to the output waveform without sequentially predicting each sample of the waveform. Originally proposed for waveform-based source separation, this architecture consists of an encoder with a series of temporal convolutions, each capturing important information at a different temporal scale to produce a low-dimensional embedding. This embedding is then decoded via upsampling operations to the dimensions of the output waveform. Skip and residual connections are present between corresponding layers of the encoder and decoder to propagate information between the two stages of operation. The WaveGAN (Donahue et al., 2019) and GANSynth (Engel et al., 2019) architectures are other examples of feed-forward networks capable of synthesizing realistic musical notes. Both architectures are based on GANs. However, AEs have also shown high potential. As we have seen in the previous section, Esling & Bitton (2018) use VAEs together

[56]https://vocaloid.com

with regularizers based on timbre perception metrics to generate instrumental sounds controllable by navigating the latent space. Bitton et al. (2019) employ adversarial AEs which are conditioned on the pitch, instrument and playing techniques for generating sounds of orchestral instruments. Several audio representations have been used in these works, such as the sound waveform (Engel et al., 2017; Donahue et al., 2019), magnitude and phase spectrograms of the signal (Engel et al., 2019) or only the magnitude spectrogram (Esling & Bitton, 2018; Bitton et al., 2019; Donahue et al., 2019; Bitton et al., 2018).

The existing work for the generation of percussive sounds is not as vast as the one for musical instruments, as these approaches do not use drum sounds in their dataset. For percussive sound synthesis using deep learning, the earliest work is the Neural Drum Machine (Aouameur et al., 2019), which uses a Conditional Wasserstein Auto Encoder (Tolstikhin et al., 2018), trained on the magnitude component of the spectrogram of percussive sounds coupled with a multi-head CNN for reconstructing the audio from the spectral representation. PCA is used on the low-dimensional representation learned by the AE to select the 3 most influential directions within the 64 dimensions of the embedding. These are provided to the user over a control interface. However these parameters controlled by the user are abstract and are not shown to be perceptually relevant or semantically meaningful. The data used for training this network was a proprietary dataset of drum sounds, which cannot be shared, making the work not reproducible.

We then proposed Ramires et al. (2020b), a feed-forward neural network architecture for the generation of drum sounds using the Wave-U-Net (Stoller et al., 2018), conditioned on high-level timbral characteristics of the sound. The conditioning features allow for reliable and intuitive control of the sound generation process by music makers while taking advantage of the fast generation process provided by the Wave-U-Net. The main shortcoming of this approach was the audio quality of the generated data, which was still not the same as professional drum samples. Together with the model, we presented a new dataset of percussive one-shot sounds, collected from Freesound, which is available to the public. The details of the model training, dataset creation and evaluation will be detailed in Chapter 9.

Two approaches based on GANs have been proposed after our work: DrumGAN (Nistal et al., 2021) and Adversarial Synthesis of Drum Sounds (ASDS) (Drysdale et al., 2020). In ASDS, the authors train a conditional Wasserstein GAN that learns to generate waveforms of drum sounds in high-resolution (44.1kHz). The conditioning signal used is a label of the drum sound to be generated, such as kick, snare or cymbal. The data that the model was trained with is also private data which cannot be shared. Despite the high resolution, some audio artifacts were still present in the generated audio and, again, we see the issue of relying on the embeddings learned by the network for controlling the generated sound. DrumGAN, on the other end, employs the same conditioning scheme as the network we proposed – the same high-level timbral features – on a ProGAN (Karras et al., 2018). The training data is Sony's in-house collection of 300 thousand drum sounds, which comprises drum sounds obtained from commercial sample pack providers (Nistal et al., 2021). The network is able to generate both the real and the imaginary component of drum spectrograms, allowing the reconstruction of the original waveform through the use of the inverse STFT. The resulting sounds are of very high quality, despite the use of 16kHz as the sampling rate. However, the coherence between the input values in the control signal and the resulting analysis of the output is lower than the one achieved by our Wave-U-Net approach. Finally, in order to generate drum loops from high-level semantic characteristics, we extended the previously proposed Wave-U-Net architecture to be conditioned on features with temporal information. This work will be presented in Chapter 10. In Chapter 11, we show our work in comparing different control parameters for percussion sound generation using GANs.

From this overview of generative models and how they can be applied to assist music creation, we can see that the most important factors for conditional waveform generation are the ability of the model to capture long-term temporal relationships of the training data, allow for intuitive manipulation of the generation process and provide audio samples with high-quality. The use of publicly available data for training the models is also essential, as it allows for reproducibility of the work and more research on the same area.

CHAPTER 9

Generation of One-shot Drum Sounds

In this chapter, we present a deep neural network-based methodology for synthesizing percussive sounds with control over high-level timbral characteristics of the sounds. This approach allows for intuitive control of a synthesizer, enabling the user to shape sounds without extensive knowledge of signal processing. We use a feed-forward convolutional neural network-based architecture, which is able to map input parameters to the corresponding waveform. We propose two datasets to evaluate our approach on both a restrictive context, and in one covering a broader spectrum of sounds. The timbral features used as parameters are taken from recent literature in signal processing. We also use these features for evaluation and validation of the presented model, to ensure that changing the input parameters produces a congruent waveform with the desired characteristics. Finally, we evaluate the quality of the output sound using a subjective listening test.

This chapter is based on **Ramires, A.**, Chandna, P., Favory, X., Gómez, E., & Serra, X. (2020). Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 786–790).

9.1 Introduction

Percussion is one of the main components in music and is normally responsible for a song's rhythm section. Classic percussion instruments create sound when struck or scraped; however new electronic instruments called drum machines were developed for generating these sounds either through playing prerecorded samples or through synthesizing them. However, these early drum machines did not provide much control over the generation of the sounds. With the developments in digital audio technology and computer music, new drum machines were hand-designed using expert knowledge on synthesis techniques and electronic music production.

With the success of deep learning, several innovative generative methodologies for creating audio have been proposed in the recent years. Neural Drum Machine (Aouameur et al., 2019) introduces percussive sound generation using deep learning, based on a Conditional Wasserstein Autoencoder (AE) (Tolstikhin et al., 2018). Principal Component Analysis (PCA) is used on the low-dimensional representation learned by the AE to select the most influential directions within the embedding space, which are given as control to users. However these parameters are abstract and are not shown to be perceptually relevant or semantically meaningful.

In our case, we wish to directly map a chosen set of features to the output sound. The Wavenet (Oord et al., 2016a) architecture has been shown to generate high quality waveforms conditioned on input features. However, the autoregressive nature of the model makes it resource extensive and the short nature of percussive sounds do not require the use of a long temporal model. Therefore, for our study, we decided to use the Wave-U-Net (Stoller et al., 2018) architecture, which has been shown to effectively model waveforms in the case of source separation and follows a feed-forward convolutional architecture, making it resource efficient. The model takes as input a waveform, downsamples it through a series of convolutional operations to generate a low dimensional representation and then upsamples it through linear interpolation followed by convolutions to the output dimensions. There are concatinative connections between the corresponding layers of the upsampling and downsampling blocks. In our work, we adapt this architecture to fit the desired use case.

The aim of our research is to create a single-event percussive-sound synthesizer that can be intuitively controlled by users, despite their sound design knowledge. This requires both a back-end of a generative model that is able to map the user controls to the output sound and a front end user interface. In this chapter, we propose a generative methodology based on the Wave-U-Net architecture (Stoller et al., 2018). Our method maps high-level characteristics of sounds to the corresponding waveforms. The use of these features is aimed at giving the end-user intuitive control over the sound generation process. We also present a dataset of ≈ 10000 percussive one-shot sounds collected from Freesound (Font et al., 2013), curated specially for this study.

The source code for our model is available online[57], as are sound examples[58], showcasing the robustness of the models.

[57]https://github.com/pc2752/percussive_synth
[58]https://pc2752.github.io/percussive_synth

9.2 Timbral features

For our end goal, we require semantically meaningful features that can allow for intuitive control of the synthesizer. In the field of Music Information Retrieval (MIR), a strong effort has been put on developing hand crafted features which can characterize sounds. These features enable users to retrieve sounds or music from large audio collections by automatically describing them according to their timbre, their mood, or other characteristics which are easy to understand by users. For our purpose, we need features pertaining to timbre. We understand timbre as pertaining to perceptual characteristics of sounds analogous to color or quality. Control over such features would enable the user to intuitively shape sounds.

A set of such features have been proposed in Pearce et al. (2017), where recurrent query terms, related to timbral characteristics, used for searching sounds in large audio databases were identified. Regression models were developed by mapping user-collected ratings to timbral characteristics, which quantify semantic attributes. These are hardness, depth, brightness, roughness, boominess, warmth and sharpness. The work proposes feature extractors pertaining to these query terms and we use an open-source implementation of the same[59]. For the rest of this chapter, we refer to the 7 features extracted by these extractors as timbral features.

Another relevant characteristic which is commonly present in drum synthesizers and music makers are used to work with is the temporal envelope of the sound. This feature describes the energy of the sounds over time and is normally available to users in drum synthesizers as a set of *attack* and *decay* controls. We use an open-source implementation of the envelope algorithm described in Zölzer (2008), present in the Essentia library (Bogdanov et al., 2013). It must be noted that the timbral features described previously are summary features, i.e. have a single value for each sound while the envelope is time evolving and of the same dimensions as the waveform.

9.3 Dataset Curation

We curated two datasets in order to train our models in different scenarios. The first consists of sounds taken from Freesound, a website which hosts a collaborative collection of Creative Commons licensed sounds[60] (Font et al., 2013). We performed queries to the database with the name of percussion instruments as keywords in order to retrieve a set of percussive sounds, with a

[59]https://github.com/AudioCommons/ac-audio-extractor
[60]https://freesound.org

limit on effective duration of 1 second. We then conducted a manual verification of these sounds[61]: to select the ones that were containing one single event, and were of appreciable quality in the context of traditional electronic music production. This process created a dataset of around 10 000 sounds, containing instruments such as kicks, snares, cymbals and bells. The dataset is publicly available in a Zenodo repository[62]. For the rest of this chapter, we refer to this dataset as Freesound One-Shot Percussive Sounds (FSOSPS).

A second dataset was created by aggregating about 5000 kick drum one-shot samples from our personal collections, originating mostly from commercial libraries. This type of sounds are often of high quality, annotated and contain only one event which makes it very handy to construct a dataset of isolated sounds, suiting our needs for training our model in a restricted context. We refer to this dataset as KICK.

The aim of creating two datasets was to understand if our method could be applicable for synthesizing a wide variety of percussion sounds, or if it was more appropriate to focus on synthesizing only one type of sounds, in this case the kick drum.

9.4 Methodology

We aim to model the probability distribution of the waveform x as a function of the timbral features fs and the time-domain envelope e. To this end we use a feed-forward convolutional neural network as a function approximator to model $P(x|fs,e)$. We use a U-Net architecture, similar to the one used by Stoller et al. (2018), which has been shown to effectively model the waveform of an audio signal. Our network takes the envelope as input and concatenates to it the timbral features, fs, broadcast to the input dimensions, as done by Oord et al. (2016a). As shown in Figure 9.1, downsampling is done via a series of convolutions with stride 2, to produce a low-dimension embedding. We use a filter length of 5 and double the number of filters after each 3 layers, starting with 32 filters. A total of 15 layers are used in the encoder, leading to an embedding of size 512. We upsample this low dimensional embedding sequentially to the output x', using linear interpolation followed by convolution. This mirrors the approach used by Chandna et al. (2019) and Stoller et al. (2018) and has been shown to avoid high frequency artifacts which appear while upsampling with transposed convolutions. As with the U-Net, there are connections between the corresponding layers of the encoder and decoder, as shown in Figure 9.1.

[61] We developed an annotation tool available at this repository https://github.com/xavierfav/percussive-annotator.

[62] https://doi.org/10.5281/zenodo.3665275

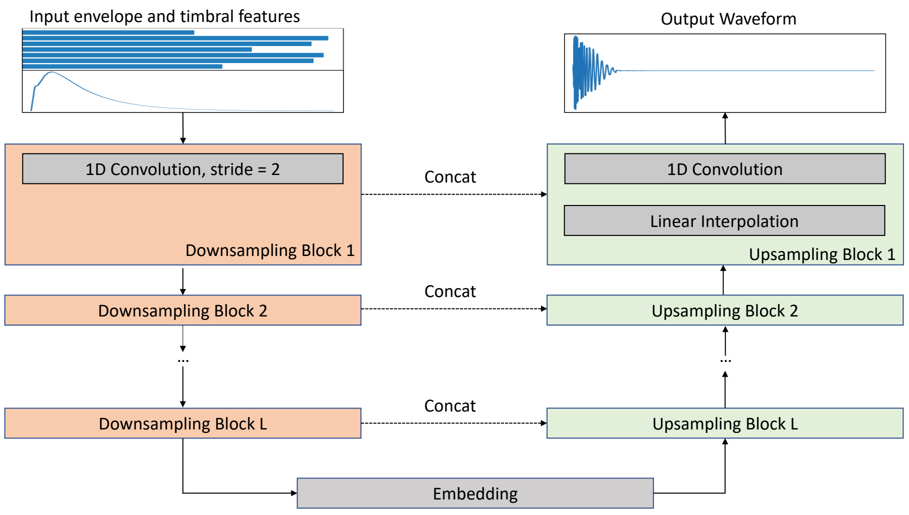


Figure 9.1: The proposed architecture, with $K = 15$ layers.

We initially used a simple reconstruction loss function, shown in equation 9.1 to optimise the network.

$$\mathcal{L}_{recon} = \mathbb{E}[||x' - x||_1] \tag{9.1}$$

While this resulted in a decent output, we noticed that the network was able to reproduce the low frequency components of the desired sound, but lacked details in high frequency components. To rectify this, we added Short-Time Fourier Transform (STFT) based loss, similar to Sahai et al. (2019). This loss is shown in equation 9.2.

$$\mathcal{L}_{stft} = \mathbb{E}[||STFT(x') - STFT(x)||_1] \tag{9.2}$$

The final loss of this network is shown in equation 9.3.

$$\mathcal{L}_{final} = \mathcal{L}_{recon} + \lambda \mathcal{L}_{stft} \tag{9.3}$$

Where λ is the weight given to the high frequency component of the reconstruction.

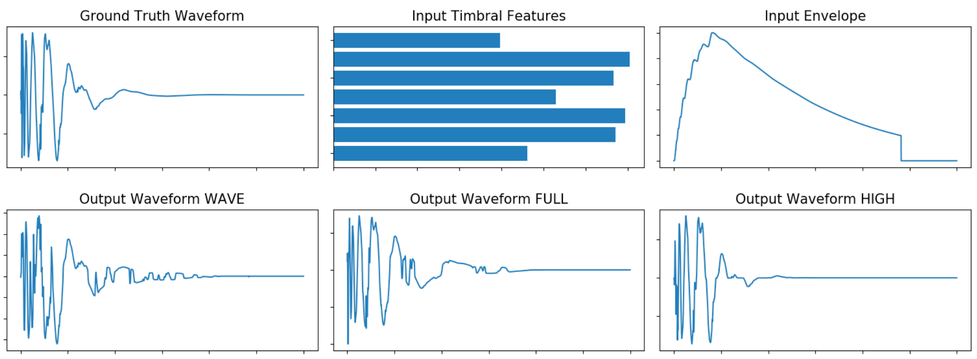


Figure 9.2: A sample of the input envelope and features and the output waveforms for the various models for the KICK dataset

9.5 Evaluation

9.5.1 Data Pre-processing

All sound were downsampled to a sampling rate of 16 kHz and silences were removed from the beginning and end of the sounds. Following this, we calculated the timbral features and envelope described in section 9.2 and then zero-padded at the end of the sound to 16 000 samples. The features were normalized using min-max normalization, to ensure that the inputs were within the range 0 to 1.

9.5.2 Network Training

The network was trained using the Adam optimizer (Kingma & Ba, 2015) for 2500 epochs with a batch size of 16. We use 90 % of the data for training and 10 % for evaluation. The STFT used for the $\mathcal{L}_{stft}$ loss function is calculated over 1024 samples and a hop size of 512. With the given sampling rate, this led to a frequency resolution of 16.125 Hz per bin. We evaluate the model with three losses: the $\mathcal{L}_{recon}$ loss, henceforth referred to as WAVE; the $\mathcal{L}_{final}$, referred to as FULL; and a version with only the high frequency components of the $STFT$ for the $\mathcal{L}_{stft}$, referred as HIGH. This last model uses $STFT$ components above 650 Hz or 40 bins as traditional kick synthesizers model a kick sound via a low frequency sinusoid, generally below 650 Hz with some high frequency noise. We use $\lambda = 0.5$ for our experiments.

9.5.3 Evaluation

The proposed models need to be evaluated in terms of the perceived audio quality and the coherence of timbral features between the input and the output. A preliminary assessment of the quality of reconstruction can be made by looking at the output waveforms, shown in Figure 9.2 for a sample from the test set of the KICK dataset. Although the reconstruction seems to be visually accurate for the three models, the perceived quality of the audio is a subjective metric that cannot be judged by simply looking at the plots. We can objectively assess the coherence of the timbral features used as input to the model. More importantly, we want to assess that a change in these features leads to a corresponding change in the output.

To this end, we vary each individual timbral feature while maintaining the other features constant. We then check the accuracy of the output waveform via the same feature extractors used for training. For each individual feature, we set values of *low*, corresponding to 0.2 over the normalized scale, *mid*, corresponding to 0.5 and *high*, corresponding to 0.8. The respective outputs for such models are termed $x'_{low}i$, $x'_{mid}i$ and $x'_{high}i$ and their corresponding features are fs^i_{low}, fs^i_{mid} and fs^i_{high} for the i^{th} feature. For coherent modeling, the models should follow the order $fs^i_{high} > fs^i_{mid} > fs^i_{low}$. We assess the accuracy of this order in three tests, $E1$, which checks the condition $fs^i_{high} > fs^i_{low}$, $E2$, which checks $fs^i_{high} > fs^i_{mid}$ and $E3$, which checks $fs^i_{mid} > fs^i_{low}$ over all values of i. The accuracy of the models over these tests is shown in Table 9.1 and a feature wise summary is shown in Table 9.2.

		Accuracy		
Dataset	Model	E1	E2	E3
FSOSPS	WAVE	0.601	0.569	0.552
	HIGH	0.649	0.601	0.657
	FULL	**0.825**	**0.758**	**0.780**
KICK	WAVE	0.805	0.722	0.722
	HIGH	0.876	0.789	0.769
	FULL	**0.920**	**0.814**	**0.798**

Table 9.1: Objective verification of feature coherence across models and datasets.

It can be seen that the FULL model, followed by the HIGH, are the most efficient at mapping the input features to the output waveform in terms of feature coherence, but all three models do maintain this coherence to a high degree.

While feature coherence is maintained for features like boominess, brightness, depth and warmth for the full dataset, the models are less consistent in terms of

	FSOSPS			KICK		
Feature	E1	E2	E3	E1	E2	E3
Boominess	0.98	0.82	0.98	0.96	0.86	0.95
Brightness	0.99	0.99	1.00	0.99	0.98	0.84
Depth	0.94	0.65	0.94	0.99	0.89	0.94
Hardness	0.64	0.66	0.59	0.85	0.61	0.79
Roughness	0.63	0.59	0.57	0.84	0.80	0.62
Sharpness	0.63	0.77	0.45	0.90	0.91	0.54
Warmth	0.92	0.79	0.91	0.88	0.61	0.87

Table 9.2: Objective verification of the accuracy on feature coherence for the best performing models for each dataset.

hardness, roughness and sharpness, particularly true for the FSOSPS dataset.

Given the absence of a suitable baseline system, we decided to use an online AB listening test that compared the models among themselves and a reference for subjective evaluation of quality. The participants of the test were presented with 15 examples each from both datasets. Each example had two options, A and B from two of the models used for the dataset, along with a reference ground truth audio. There were 5 examples each from each of the 3 pairs. The participant was asked to choose the audio clip which was closest in quality to the reference audio. There were 35 participants in the listening test, the results of which are shown in Figure 9.3.

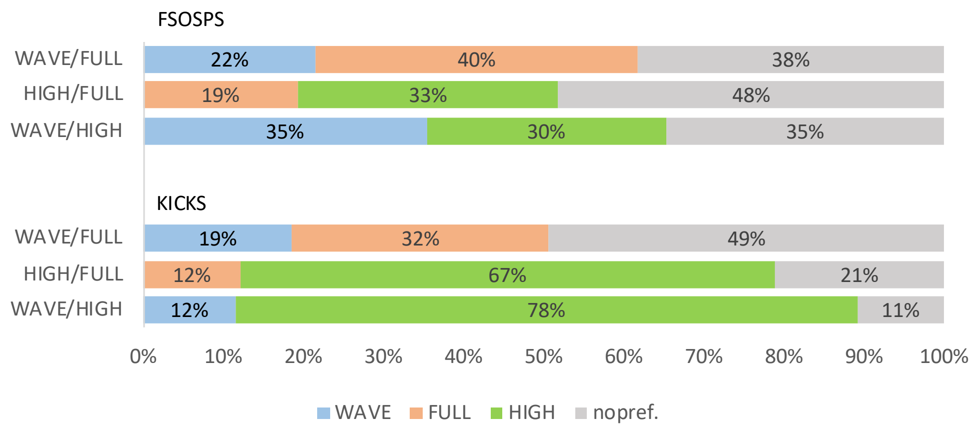


Figure 9.3: Results of the listening test, displaying the user preference between loss functions for each of the datasets.

A clear preference for the HIGH model can be seen, especially for the KICK dataset. This can be attributed partly to the choice of cutoff frequency used in the model and partly to the diversity of sounds in the FSOSPS dataset. We note the difficulty in assessing audio quality over printed text and encourage the user to visit our demo page and listen to the audio samples for assessment.

9.6 Conclusions and Future Work

In this work, we proposed a method using a feed-forward convolutional neural network based on the Wave-U-Net (Stoller et al., 2018) for synthesizing percussive sounds conditioned on semantically meaningful features.

Our final aim is to create a system that can be controlled using high-level parameters, being semantically meaningful characteristics that correspond to concepts casual music makers are familiar with. To this end, we use hand crafted features designed by MIR experts and curate and present a dataset for the purpose of modeling percussive sounds. Via objective evaluation, we were able to verify that the control features do indeed modify the output waveform as desired and quality assessment was done via an online listening test.

Future work on this area involves developing an interface for interacting with the synthesizer, which allows to evaluate the approach in its context of use, with real users. Improvements in the generated sound quality and extending this architecture to be able to generate percussive loops will be the focus of the following chapters.

CHAPTER 10

Generation of Drum Loops

In this chapter, we present LoopNet, a feed-forward generative model for creating loops conditioned on intuitive parameters. As a continuation to the previous chapter, we use a large collection of public loops with the Wave-U-Net architecture to map control parameters to audio. We also evaluate the quality of the generated audio and propose intuitive controls for composers to map the ideas in their minds to an audio loop.

This chapter is based on Chandna, P., **Ramires, A.**, Serra, X., & Gómez, E. (2021). LoopNet: Musical Loop Synthesis Conditioned on Intuitive Musical Parameters. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 3395-3399).

10.1 Introduction

The concept of loops is not new, repetitive units or motifs which are repeated through musical compositions have long been used in music across cultures, allowing composers to maintain continuity with various degrees of variability and complexity.

Audio synthesis technologies for music have been researched for many years, ranging from synthesizers generating pitched waveforms, to singing voice synthesizers conditioned on melody and text[63], to deep learning based models capable of generating entire songs (Carr & Zukowski, 2018; Zukowski & Carr, 2018). Initially proposed for generating high-quality speech samples, the Wavenet architecture models each sample of a waveform as a function of the previously predicted time steps, leading to the autoregressive nomenclature. The use of dilated causal convolutions allows the architecture to model longer term temporal dependencies between samples in an audio waveform than in the

[63]https://vocaloid.com

SampleRNN architecture. This architecture has subsequently been adapted for musical generation like singing voice synthesis conditioned on lyrics (Blaauw & Bonada, 2017), and instrument sound generation conditioned on the pitch and latent representations of timbre (Engel et al., 2017; Dieleman et al., 2018; Carr & Zukowski, 2018). While the output of these models is subjectively similar to natural-sounding samples, the sequential nature of the model means that the processing time for generation is quite high, unless high-resource processing units are available. Feed-forward adaptations of the architecture via probability density distillation have been proposed showing that it is possible to directly map the input conditioning to the output waveform without sequentially predicting each sample of the waveform (Oord et al., 2018).

Efforts have been made to find semantic structures within the latent embeddings that can provide an intuitive control to the music producer, but there remains a gap between the intuition behind the parameters of the generative model and the perceptual qualities of the audio generated. To this end, works have used Music Information Retrieval (MIR) techniques to generate single shot percussive sounds conditioned on semantically relevant musical features that are perceptually easier for the user to understand and manipulate (Ramires et al., 2020b; Nistal et al., 2021). In this chapter, we aim to extend the generation of sounds conditioned on perceptually relevant features from one-shot sounds to loops.

To this end, we model the waveform of a loop as a function of global features pertaining to timbre and harmony and time-varying features pertaining to rhythm. While there are several deep learning based architectural choices (Dieleman et al., 2018; Donahue et al., 2019; Mehri et al., 2017) for modeling such a distribution, we decided to use a feed-forward convolutional design with skip and residual connections for the propagation of information between the layers of the network. This allows for the feed-forward generation of waveforms conditioned on the input. We leverage a curated collection of loop samples for training the model and evaluate our proposed methodology in terms of the perceived quality of the generated audio, the processing time required and the feature coherence between the input conditioning and the output sample. We also release our code publicly for reuse and provide interactive examples for a demonstration [64].

The rest of the chapter is structured as follows: Section 10.2 presents a brief overview of the dataset we use for training and testing our model. Our proposed methodology is outlined in Section 10.3 and a description of the evaluation strategy we use is presented in Section 10.4. This is followed by the results of our study in Section 10.5 and a discussion of the work presented here and how it can be continued is presented on Section 10.6.

[64]https://github.com/aframires/drum-loop-synthesis

10.2 Dataset Curation and Analysis

To collect a dataset for training the generative model, we use loops from an in-house collection of loops from Looperman[65], a community loop database. This database contains loops provided by users with annotations for genre, instrumentation, key and tempo.

We collected 8838 loops from the *drums*[66] category, with tempo annotations ranging from 120 to 140 Beats Per Minute (BPM), the most frequent tempos in the collection. As tempo annotations provided by users might be noisy, we validate the tempo using a confidence measure (Font & Serra, 2016), taking into account the difference in the duration of a single bar and the relative duration of the entire loop. We only use loops which have a confidence measure higher than 99%, leading to a total of 8226 loops.

The Wave-U-Net architecture works with fixed-length input and output, requiring all loops to have the same length. To this end, we use the rubberband library[67] to time-stretch all the loops to 130BPM. We verified that this time-stretch did not create artifacts and that the loops were still sounding coherent after this step. To feed our model with fixed-length input, we split the loops into 1-bar segments, with a duration of 1.846 s at 130BPM. This process led to 49 045 1-bar segments, which we then downsampled to 16 kHz and analyzed as shown in the next section.

We used 90% of the 1-bar loops for training and the rest for testing. As segments from the same loop are likely to be similar, we performed the training-test split before segmenting the audio into 1-bar segments. As a result, there is no overlap between the loops from which segments were used for training and those which were used for testing.

10.3 Methodology

Once pre-processed, we analyze the loops to extract perceptually relevant features which we map back to the waveform via the neural network. We consider two types of features: local time-varying features, which are defined over the same temporal scale as the output and global features which can be summarized to be consistent across the length of the loop.

[65] https://looperman.com

[66] The instrumentation in the loops is not restricted to drums, even though the keyword is used for shortlisting. The loops might also include complementary melodic instruments on top of the drums.

[67] https://breakfastquay.com/rubberband

10.3.1 Time-varying Conditioning Features

Local time-varying features correspond to the rhythm of the loop. To obtain information on a loop's rhythmic pattern, we use an Automatic Drum Transcription algorithm (Southall et al., 2017). This algorithm models the probability of a windowed frame of the analyzed audio having a kick drum, a snare drum or a hi-hat. We use one-dimensional spline interpolation [68] to interpolate the values from the windowed frames to the length of the waveform. An example of this representation for a loop in our dataset is presented in Figure 10.1. As seen in the figure, the activation function has sparse energy distribution across time, which does not represent the energy distribution of the loop. To account for this, we also calculated the envelope of the waveform, as in the previous chapter.

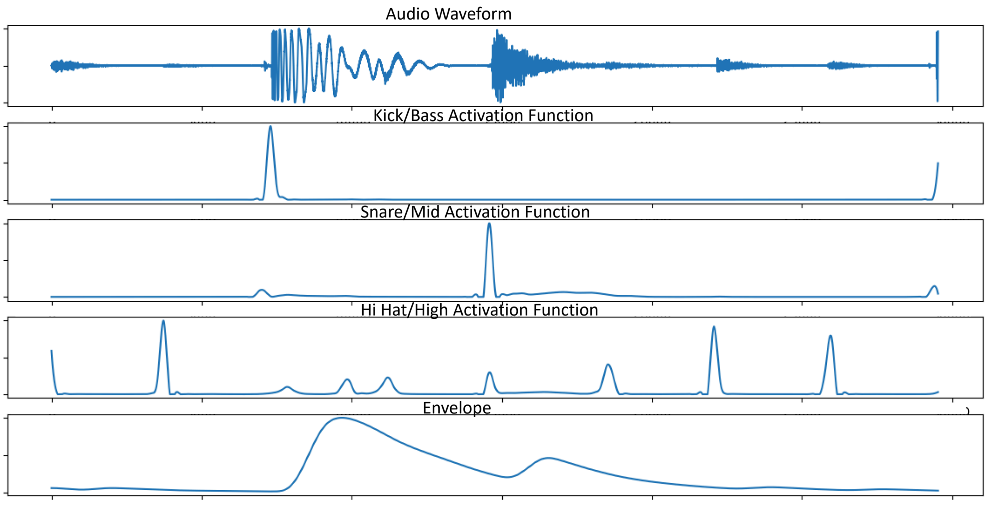


Figure 10.1: The waveform of the audio along with the activation function extracted using the Automatic Drum Transcription algorithm and the energy envelope.

10.3.2 Global Conditioning Features

Global conditioning features correspond to textural features, both harmonic and non-harmonic, which can be assumed to be constant through the duration of the loop. To summarize the harmonic texture of the audio across the length

[68]https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.interpolate.spline.html

of the loop, we used Harmonic Pitch Class Profiles (HPCP) (Gómez, 2006b)[69]. This feature represents the energy distribution across the 12-note chromatic scale commonly used in western music and is also intuitive for music makers. To model the abstract non-harmonic texture of the loop, we used the perceptually pertinent timbral features proposed by Pearce et al. (2017). These are hardness, depth, brightness, roughness, boominess, warmth and sharpness[70]. We used the methodology proposed by Miron et al. (2013), to separately analyze the different frequency bands in each loop. The signal was analyzed through three different filters; a 1st order IIR low-pass filter with a cutoff frequency of 90 Hz, a 2nd order IIR band-pass centered in 280 Hz, and a 1st order IIR high-pass filter with cutoff frequency at 9000 Hz. The global conditioning feature for each of the frequency bands was summarized by taking an average across all windowed frames, concatenated, normalized and broadcast across time for conditioning the network as in Oord et al. (2016a).

10.3.3 Architecture

We use the feed-forward Wave-U-Net architecture (Stoller et al., 2018), used in the previous chapter, for mapping the input features to the output waveform, x. The architecture is shown in Figure 10.2, and consists of an encoder and a decoder. The encoder downsamples the input via a series of convolutions with stride 2, to produce a low-dimension embedding. A filter length of 5 is used throughout the layers and the number of filters is doubled after each 3 layers, starting with 32 filters. We use 10 layers in the encoder, to produce a low-dimensional embedding. This embedding is upsampled via linear interpolation (Chandna et al., 2019; Stoller et al., 2018) and each upsampling operation is followed by a 5×1 convolution to generate the output, $\hat{x}$. The inputs and outputs are vectors of length 29 538, representing the number of samples in 1 bar loops in our dataset. The corresponding layers in the encoder and decoder are connected via the concatenation of features to allow for the propagation of information (Stoller et al., 2018).

10.3.4 Loss Functions

In this work, we experiment with several loss functions and compare their output quality. We used the reconstruction loss shown in Equation 10.1 and complemented it with a perceptually motivated loss based on the Short-Time Fourier Transform (STFT) based loss, shown in Equation 10.2. We used a hop

[69]The Essentia implementation at https://essentia.upf.edu/reference/streaming_HPCP.html with the default parameters was used.

[70]To obtain these features we used an open-source implementation https://github.com/AudioCommons/ac-audio-extractor.

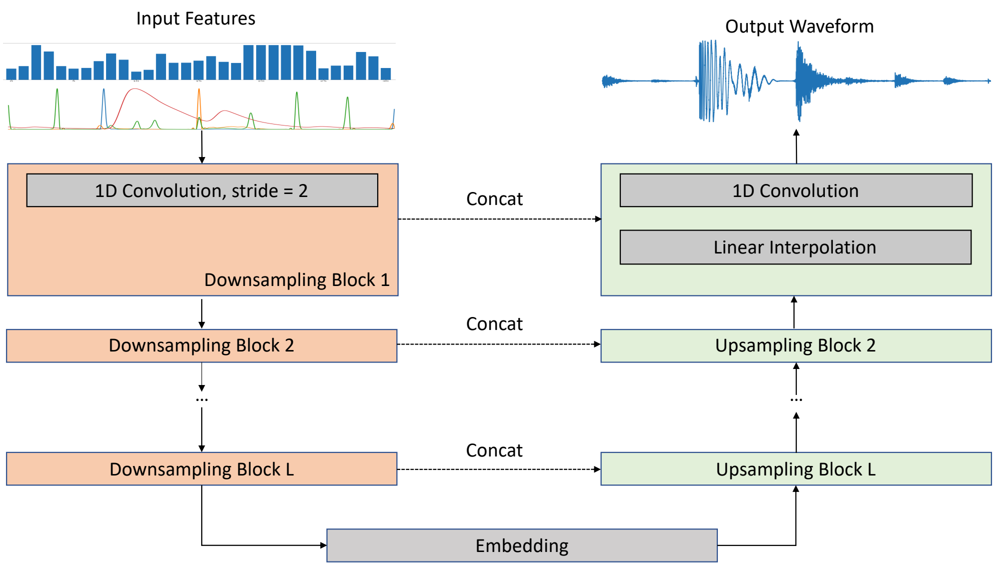


Figure 10.2: The Wave-U-Net architecture used in our study, the input includes the local conditioning with onset detection functions, the global HPCP and timbral features which are broadcast along time. The output is the waveform of the loop.

size of 512 for calculating the STFT for $\mathcal{L}_{stft}$, over a STFT window resolution of 1024 samples, resulting in a frequency resolution of 16.125 Hz per bin. In addition, we used the multi-resolution loss (Wang et al., 2020; Engel et al., 2020), which consists of calculating the STFT with various FFT window resolutions (2048 samples for $i = 0$, 1024 samples for $i = 1$, 512 samples for $i = 2$, 256 samples for $i = 3$, 128 samples for $i = 4$, 64 samples for $i = 5$).

$$\mathcal{L}_{recon} = \mathbb{E}[||\hat{x} - x||_1] \tag{10.1}$$

$$\mathcal{L}_{stft} = \mathbb{E}[||\hat{x} - x||_1] + \mathbb{E}[||STFT(\hat{x}) - STFT(x)||_1] \tag{10.2}$$

$$\mathcal{L}_{multi} = \mathbb{E}[||\hat{x} - x||_1] + \sum_{i=0}^{5} \mathbb{E}[||STFT_i(\hat{x}) - STFT_i(x)||_1] \tag{10.3}$$

For the rest of the chapter, we will refer to the model optimized using Equation 10.1 as *WAV*, the model optimized for Equation 10.2 as *WAVSPEC* and that optimized using the multi-resolution loss (Engel et al., 2020) shown in Equation 10.3 as *MULTI*.

10.4 Experiments

10.4.1 Models

In our experiment, we train and evaluate 5 different models. The first of the models, termed as *STFT*, maps control features to the STFT of the waveform and uses the Griffin-Lim (Griffin & Lim, 1984) algorithm for waveform reconstruction. The *WAV*, *WAVSPEC* and *MULTI* models map the input features directly to the waveform and are optimized using the loss functions described in Section 10.3.4. In addition, we train a model mapping the input features minus the envelope to the corresponding waveform, optimized using the multi-resolution loss. We term this model as *MULTI NOENV*.

10.4.2 Parameters

The network was trained using the Adam optimizer (Kingma & Ba, 2015) with a batch size of 16. Like Dieleman et al. (2018), we found that after a certain limit of optimization, the loss function did not directly correspond to the audio quality of the output. As such, we heuristically selected the best epoch for synthesis for each of the models.

10.4.3 Evaluation

While there are several aspects of the generated output that can be assessed, we restrict our evaluation to two aspects: the audio quality and the coherence between changes in the input features to the output audio.

10.4.3.1 Audio Quality Assessment

Audio quality is a subjective attribute, which is difficult to quantitatively measure as each person listening to the audio is likely to have their own perception of the quality. Some quantitative metrics based on deep learning have been proposed recently to assign a quantitative degree to this attribute, including the *Fréchet Audio Distance (FAD)* (Kilgour et al., 2019), the *Inception Score* (Salimans et al., 2016) and the *Kernel Inception Distance (KID)* (Bińkowski et al., 2018). In this study, we use the FAD between the original test set and the generated output to assess the audio quality of the output. This metric can identify degradation of audio quality in a way related to human judgments, using audio-based latent embeddings from the Audioset (Gemmeke et al., 2017) VGG'ish pre-trained model[71]. To create a baseline for this evaluation, we re-

[71] https://github.com/tensorflow/models/tree/master/research/audioset

synthesized the audio using the Griffin-Lim (Griffin & Lim, 1984) algorithm. This baseline is termed as *Griffin-Lim* and has minimal degradation.

10.4.3.2 Timbral Feature Coherence

We want changes in the input features to be reflected in the synthesis output. To this end, we assess feature coherence over 16 loops from the test set in a manner similar to Ramires et al. (2020b) and Nistal et al. (2021). We synthesized outputs changing each of the 21 timbral features individually, while keeping the rest unchanged. The feature was changed to three different values: 0.2, 0.5 and 0.8, over the normalized scale. We refer to this outputs as $x'_{low}i$, $x'_{mid}i$ and $x'_{high}i$ and their corresponding features are fs^i_{low}, fs^i_{mid} and fs^i_{high} for the i^{th} feature. This resulted in $16 \times 21 \times 3 = 1008$ different outputs for each model. We then extracted timbral features as described in Section 10.3.2.

We evaluate timbral feature coherence by validating if the models follow the order $fs^i_{high} > fs^i_{mid} > fs^i_{low}$. For this, we use three tests (Ramires et al., 2020b; Nistal et al., 2021): $E1$, which checks the condition $fs^i_{high} > fs^i_{low}$; $E2$, which checks $fs^i_{high} > fs^i_{mid}$; and $E3$, which checks $fs^i_{mid} > fs^i_{low}$ over all generated loops.

10.5 Results and Discussion

10.5.1 Audio Quality Assessment

The results for the assessment of audio quality using the FAD metric are presented in Table 10.1. We encourage the reader to listen and subjectively evaluate the outputs for each model, provided in the accompanying website[72]. The website includes synthesis examples from the test set, as well as an illustration of user-interaction possibilities like timbre transfer

As seen in Table 10.1, the spectogram loss in the *WAVSPEC* model leads to an improvement in synthesis quality over the *WAV* and *STFT* models. The use of the multi-resolution loss in the *MULTI* models led to a significant improvement in audio quality measured by the FAD. We also see that the use of the overall envelope was redundant as the model was able to perform just as well when not conditioned on this feature. We stress that this measure is just a simplification of a highly subjective perceptual attribute and acknowledge that there is room for improvement in the audio quality.

[72]https://github.com/aframires/drum-loop-synthesis

Model	FAD
Griffin-Lim	1.26
STFT	24.97
WAV	13.83
WAVSPEC	9.06
MULTI	3.73
MULTI NOENV	3.35

Table 10.1: FAD for the outputs of each of the models, when compared to the original test data. FAD values closer to 0 indicate higher similarity between the quality of the original audio and the assessed output.

10.5.2 Timbral Feature Coherence

The results of the feature coherence evaluation are presented in Table 10.2. A high degree of coherence can be observed between changes in input features and the resulting output for all models except the *WAV* model. The *STFT* model outperforms the waveform based models for feature coherence but lags in audio quality.

	Accuracy		
Model	E1	E2	E3
STFT	87.50%	77.68%	79.46%
WAV	67.44%	62.86%	52.14%
WAVSPEC	76.79%	75.89%	74.11%
MULTI	87.50%	77.68%	75.00%
MULTI NOENV	83.93%	77.68%	75.00%

Table 10.2: Mean feature coherence across models for each error type.

10.6 Conclusion

We present LoopNet, a deep learning-based feed-forward loop synthesis algorithm, mapping intuitive input controls directly to the corresponding waveform. The nature of the model allows for fast synthesis of the loop, with processing time from control input to the synthesized output of the order of 0.5 s per loop, without the use of a GPU. In addition to the models presented in this study, we tried and tested various other loss functions, including training via the adversarial optimization and using DDSP (Engel et al., 2020) features for synthesis. In this study, we present some of the handpicked models which worked the best. We have verified that the models can maintain feature coherence between the control inputs and the generated output. The quality of the sound generated is a subjective aspect, that we have simplified and evaluated using a deep learning based methodology. The results are encouraging but there is room for improvement.

The control features presented provide innovative avenues for user control over the synthesis. Apart from the straightforward mapping of MIDI input over a time grid to the input rhythm features, the drum transcription function can also be computed directly from audio, allowing for mixing and matching of rhythm, timbre and harmonic features over loops or even over more abstract sounds such as human beatboxing or environmental sounds with interesting timbre. We provide examples of these on our complimentary website along with other interfaces for input control. We believe this will allow for a greater degree of freedom for musicians looking to expand their musical creativity while using loops for compositions. This also establishes a baseline for future works in loop synthesis looking to improve the quality of the loops synthesized.

CHAPTER 11

Comparing Representations for Drum Synthesis

In the previous chapters, we have proposed methodologies for controlling the generative process of drum sounds using high-level semantic feature parameterization. Different works have proposed different interfaces but no comprehensive analysis has been presented to evaluate how each approach relates to the creative process. In this chapter, we evaluate how different interfaces support creative control over drum generation by conducting a user study based on the Creativity Support Index (CSI). We experiment with both a supervised method that decodes semantic latent space directions and an unsupervised Closed-Form Factorization approach from computer vision literature to parameterize the generation process.

This chapter is based on **Ramires, A.**, Juras, J., D. Parker, J., & Serra, X. (2022). A Study of Control Methods for Percussive Sound Synthesis Based on GANs. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

11.1 Introduction

Sound synthesis techniques for percussion sounds have evolved significantly throughout the last decades, with many techniques being associated with and even defining entire Electronic Music genres. Analogue synthesis paved the way for creating and designing percussive sounds electronically. More recently, with the development of music-making software such as Kick2[73], SubLab[74] and the default drum synthesizers available in Digital Audio Workstations, digital drum sound creation became common practice for music makers of

[73]https://sonicacademy.com/products/kick-2

[74]https://futureaudioworkshop.com/product/sublab

all backgrounds, enabling advanced digital signal processing techniques to be applied to drum sound design.

Recent advances in deep learning introduced novel methodologies for synthesizing data. Architectures such as Autoregressive Networks (Oord et al., 2016a; Engel et al., 2017; Mehri et al., 2017), Variational Autoencoder (VAE) (Kingma & Welling, 2014), and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have all been proven to generate high-quality results in a variety of domains, from images of human faces to musical audio. Besides achieving the best synthesis quality in several domains, recent work has shown that GANs can even capture high-level semantic concepts (Yang et al., 2021) in the latent space dimensions driving the networks. Nevertheless, determining the correct latent space feature to manipulate to achieve a specific variation in the data space can be cumbersome – especially when dealing with high-dimensional latent spaces. To overcome this issue, new techniques have been proposed to find directions in the latent space that correspond to semantic concepts, either in a supervised (Shen et al., 2020) or unsupervised manner (Shen & Zhou, 2021).

Recently, research into percussive sound creation using generative deep learning models has been receiving increased attention. Both DrumGAN (Nistal et al., 2021) and Adversarial Synthesis of Drum Sounds (ASDS) (Drysdale et al., 2020, 2021) employed the GAN training paradigm for this task. Each proposed its own methodology for controlling the synthesis, conditioning on timbral features, and the drum class respectively. Despite the significant research effort in this area, no studies have yet compared the various approaches made to controlling the synthesis process itself.

The main goal of our research is to evaluate 3 different methodologies for navigating the latent space of a GAN trained to generate drum sounds. To this end, we adapted StyleGANv2 (Karras et al., 2020) to accommodate the dimensionality of the time-frequency representation of musical audio and trained the network on a private dataset of drum sounds. Both a supervised (Shen et al., 2020) approach – where features determined from data generated by the trained network are used to infer synthesis control – and an unsupervised (Shen & Zhou, 2021) method – where synthesis control is determined based on the learned weights of the network itself – were applied to the trained network to find perceptually salient directions in the latent space. Finally, a user study based on the CSI (Cherry & Latulipe, 2014) was employed to compare these two latent navigation approaches against a simplistic approach to latent space vector manipulation.

11.2 StyleGAN2

As its name implies, StyleGAN2 is a flavor of GAN – a neural network architecture that exploits adversarially training independent generator (G) and discriminator (D) networks. The input to the former are one-dimensional vectors $\mathbf{z}$ from a latent space Z - an independent and identically distributed Gaussian probability distribution, while the latter is input with one-, or multi-dimensional, vectors of data $\mathbf{x}$ from the data space X, which represents all real data instances. The generator is tasked with learning a mapping between $p_z(z)$, and $p_{data}(x)$, the probability distribution of all training data samples. The discriminator is a classifier that ideally scores real data examples (training data) with a score of $D(X) = 1$, and generated data examples with a score of $D(G(Z)) = 0$. Thus, the discriminator wishes to maximize the probability of assigning the correct label to both training data and generated data. For training data, it is trained to maximise the expected value over all instances in X:

$$\mathbb{E}_{X \sim p_{\text{data}}}[logD(X)] \tag{11.1}$$

Likewise for generated data, the discriminator is trained to maximize the expected value over all generated fake instances $G(Z)$:

$$\mathbb{E}_{Z \sim p_z}[log(1 - D(G(Z))] \tag{11.2}$$

while the generator works to minimize Equation 11.2. Adversarial training amounts to a *two-player minimax* game between the generator and discriminator networks:

$$\begin{aligned}\min_G \max_D V(D,G) := \mathbb{E}_{X \sim p_{\text{data}}}[\log D(X)] + \\ \mathbb{E}_{Z \sim p_z}[\log(1 - D(G(Z))]\end{aligned} \tag{11.3}$$

StyleGAN2 is part of a lineage of generative models developed by the NVIDIA research team. Initially motivated by stabilizing the training process for GANs, Karras et al. (2018) proposed ProGAN – a new model architecture and corresponding training procedure that 'progressively' trained layers of a deep convolutional GAN against different downsampled resolutions of its training data.

StyleGAN radically revised the deep convolutional GAN architecture, by redefining the functional relationship between latent space vectors and the generator network (Karras et al., 2019): (i) As a means to disentangle possible non-linear subspaces within the normally distributed latent space $\mathbf{z} \in Z$, a learned intermediate latent space, or *style space*, $\mathbf{w} \in W$, was introduced. Connected to the latent space by a non-linear mapping, $f : Z \to W$, the style space doesn't have to support sampling according to any fixed distribution; (ii) Instead of feeding the latent vector directly to the generator network like in ProGAN, StyleGAN fed the generator with a fixed seed and applied the style vectors in **w** across each layer of the generator through an affine transformation. This effectively applied the affine-transformed style to each level of resolution in the network, influencing coarse features at lower resolution layers, and fine-grained features at higher resolution layers.

Improving on ProGAN's unsatisfactory performance in generating stochastic image features (hair, background foliage, pores, etc.), StyleGAN further introduced noise to each resolution layer of the generator, scaled by a learned weight. While the network generated state-of-the-art images, artifacts from the training procedure – notably shift-invariance, were left as open questions for future work.

StyleGAN2 introduces several improvements on the original StyleGAN architecture (Karras et al., 2020). To address 'blob'-like artifacts that were common in generated StyleGAN images, StyleGAN2 replaces inter-layer normalization (Adaptive Instance Normalisation (AdaIN), which independently normalizes both the mean and variance between adjacent convolution layers) with what Karras et al. refer to as weight demodulation. In general, the goal of inter-layer normalization is to remove the statistics of the applied style vector, **w**, from the output feature map. However, by normalizing both the mean and variance between layers, information discovered by the network about the magnitudes of the features relative to each other is potentially destroyed – which is speculated as a culprit for the 'blob'-like artifacts. Weight demodulation is proposed as a 'weaker' means to normalize than AdaIN (and respectively pixel-wise normalization in ProGAN), since it is based on statistical assumptions of the signal passing through the layer rather than the actual contents of the feature map – which thus preserves relative magnitude information between layers.

Furthermore, the StyleGAN2 network is no longer trained progressively: it was shown that with a large enough training dataset, the gradient updates applied to the network during training are roughly in line with how ProGAN trains – ie. early training focuses on lower resolution layers, and progressively fine-tunes higher resolution layers; and without the risk of shift-invariant image generation.

The many innovations of StyleGAN2 led us to believe that it would be a well-suited architecture to synthesize drum sounds. Notably, given the many different drum classes present in our dataset, the disentangled nature of the network architecture's *style space* presents the potential for latent conditioning suitable for coherent interpolation between drum classes. Furthermore, the trainable stochastic noise components fed to each network layer are well suited for the task of generating the noise components common to drum sounds – from kick drum transients to sustained hi-hats and cymbals.

11.3 Controlling the Generation

To allow a degree of control over the synthesis process in generative models, several approaches have been proposed. Previous research on percussive sound generation used conditioning features which, based on an external conditional signal c, force the model to learn the conditional probability $p(x|c)$. The chosen conditioning signal c can vary in terms of how much information it contains, from low information signals like class labels (e.g. kick, snare or cymbal), to very rich conditioning signals like the envelope of the drum sound.

In this chapter we compare supervised and unsupervised approaches for finding latent directions in the learned latent space of GANs. While conditioning can be used as a control for the generation process, we want to create a fair comparison between approaches. We therefore focus on approaches that do not require constraining the network during training and can be applied directly to a pre-trained model. Our goal is to find latent directions $n \in \mathbb{R}^d$, with some interpretable meaning, which allow the modification of a sound $G(z)$ with latent code z to a new sound $G(z') = G(z + \alpha n)$ where α represents the amount of modification.

Unsupervised methods rely on applying dimensionality reduction to the trained latent space to find the directions n that correspond to the most significant change in the output. Early experiments (Härkönen et al., 2020) relied on generating data from points in the latent space and posterior application of PCA to discover the directions. In this work, we use Semantic Factorization (SeFa) (Shen & Zhou, 2021), a closed-form factorization method that does not require sampling and can learn directions directly from the weights of the trained model. The paper shows that given any latent code z, and the weight matrix A, the edit operation $G(z')$ can be achieved by adding the term αAn on the projected code. Therefore, A contains all information related to the output variation. The basis for finding the latent directions in SeFa is to solve the optimization problem:

$$n = \underset{n \in \mathbb{R}^d : n^T n = 1}{\arg\max} \; ||An||_2^2 \tag{11.4}$$

where $||.||_2$ denotes the l_2 norm. The solutions for this problem are shown to be the eigenvectors of $A^T A$ with the largest eigenvalues. This method showed remarkable performance when applied to the pre-trained StyleGAN (Karras et al., 2019) model for generating faces, being able to identify the directions corresponding to pose, presence of glasses, gender and amount of smiling, in a more disentangled manner than PCA.

Supervised methods for discovering latent space directions require an annotation procedure on synthesized data to train classifiers in the latent space. In

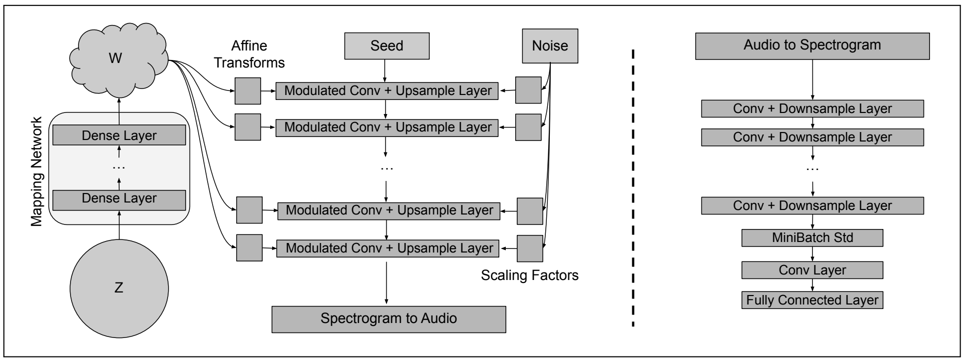


Figure 11.1: StyleGAN2 Generator (left) and Discriminator (right) architectures.

InterfaceGAN (Shen et al., 2020), it is assumed that, for binary characteristics, there is a hyperplane in the latent space which separates positive and negative examples. To find the hyperplane, a large amount of data needs to be generated from the trained network to be later classified using classification algorithms. The authors experimented with classifiers for the pose, smile, age, gender and eyeglasses to get positive and negative labels for the generated data and used Support-Vector Machines (SVMs) to then find the dividing latent space hyperplane for each characteristic. The direction n that encodes a characteristic to modify is therefore a normal vector of the discovered hyperplane, which passes by, z, the latent code we want to modify. It is also shown that the larger the magnitude of the modification α is, the more affected the sample is according to the encoded direction – despite n being found through a binary classification hyperplane. This algorithm also has shown remarkable performance when editing faces in terms of pose, smile and age.

11.4 Methodology

11.4.1 Dataset

The network was trained on the entire corpus of one-shot drum samples included with Native Instrument Maschine Expansion releases. Table 11.1 shows the distribution of sample counts for each drum class in the dataset. These samples were all created by the Native Instruments sound design department throughout the last decade. As a result, there is an inherent consistency across the dataset in terms of quality, onset locations, and pre-normalized sample volumes.

Drum Class	**Count**	**Drum Class**	**Count**
Claps	1547	Combo	91
Cymbal	1270	Hand Drum	99
HiHat	4645	Kick	4025
Mallet Drum	5	Metallic	73
Percussion	3365	Shaker	1122
Snare	3332	Tambourine	3
Tom	1017	Wooden	36
		Total	**20630**

Table 11.1: Distribution of drum classes in the training dataset.

11.4.2 Data Pre-processing

As our network was trained on a two-dimensional spectrogram representation of our audio data sampled at 16kHz, data pre-processing was implemented as follows. (i) Audio recordings were resampled to 16kHz and zero-padded to 16k samples, representing one second of audio data. (ii) A logarithmic 'fade-out' was added to the last 30% of each audio vector. (iii) Audio vectors were normalised to a floating-point range of [-1.0, 1.0]. (iv) Audio vectors were converted to complex spectrograms, using the following parameters: hop size of 512 samples, window size of 2048 samples, and an FFT size of 2048 samples. (v) Complex valued spectrogram reshaped into a 2 channel feature map of real and imaginary components per channel. (vi) Finally, the DC component of the 2 channel spectrogram representation was removed.

11.4.3 Model and Training

The default implementation of StyleGAN2 provided by NVIDIA was used, with some adaptations made to it to work with spectrograms of shape 1024×32: (i) The network was modified to handle rectangular shapes instead of only square data. (ii) The resolution of the smaller feature map in the generator was set to 64×2, which is doubled every layer. (iii) The network was adapted to handle only 2 channels, instead of the 1 or 3 channels commonly used for image generation. We use 5 synthesis blocks in the generator comprising a Modulated Convolutional and an Upsampling layer. On the discriminator 5 blocks comprising of a Convolutional and a Downsample layer are used. An overview of the complete network is presented in Figure 11.1.

The network was trained on a virtual Google Cloud Platform machine, using PyTorch's GPU library *Distributed Data Parallel* to train across 4 NVIDIA Tesla T4 GPUs. The latent space and style space dimensions were both set to 512, and the learning rate was set to 2e−3. A batch size of 8 examples was

Feature	Validation Accuracy	Test Accuracy
Boominess	100%	70.9%
Brightness	99.2%	67.6%
Depth	99.2%	72.7%
Hardness	97.5%	72.0%
Roughness	100%	52.3%
Sharpness	100%	66.5%
Warmth	100%	81.8%

Table 11.2: SVM accuracy when separating positive and negative examples in InterfaceGAN.

used. Although the stated GAN training objective is to arrive at an equilibrium point, where the discriminator outputs similar scores for real and generated data, in practice, the quality of data output from the network is typically maximized before the equilibrium is reached. A Fréchet Audio Distance (FAD) analysis was used to determine which training epoch corresponds to the highest quality and most diverse audio output (Kilgour et al., 2019). Epoch 243, which corresponds to the network being exposed to 5,012,000 spectrograms, generated a minimum FAD score of 2.689. The code used to train and create the model, as well as audio examples for the reader to assess the audio quality are available on the accompanying website.[75]

11.4.4 User Interface

We want to evaluate how our drum generation model can help foster creativity when assisting music makers in creating drum sounds. To this end, we created a graphical user interface that allows generating random sounds, navigating the Z latent space, and also modifying sounds according to the directions learned by the SeFa and InterfaceGAN algorithms. The interface can be seen in Figures 11.2 and 11.3.

The first panel on the left of the user interface in Figure 11.2 represents the Z latent space. This is the first interface we evaluate. The user can set the value for each of the 512 dimensions by either drawing the vector with the computer mouse, or randomly seeding each latent dimension, and generate the corresponding audio output.

The second interface and control methodology we want to evaluate are the 7 most significant directions returned by the unsupervised SeFa algorithm when applied to our trained model. The choice of 7 as the number of directions was to have the same number of control parameters as the timbral features used for

[75] https://aframires.github.io/stylegan2-ada-pytorch

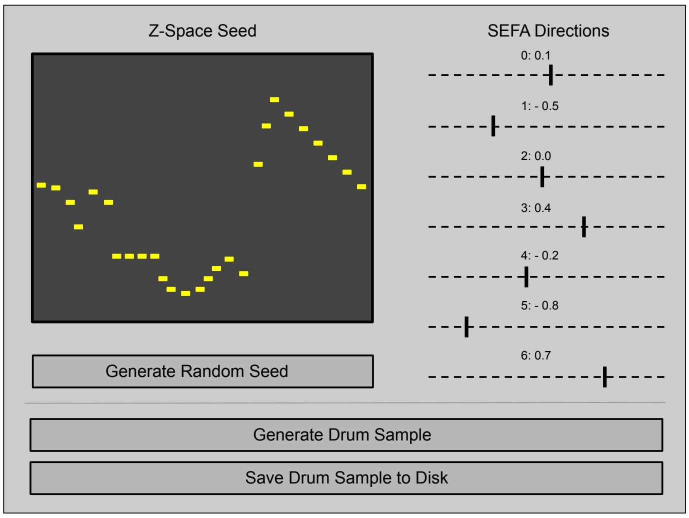


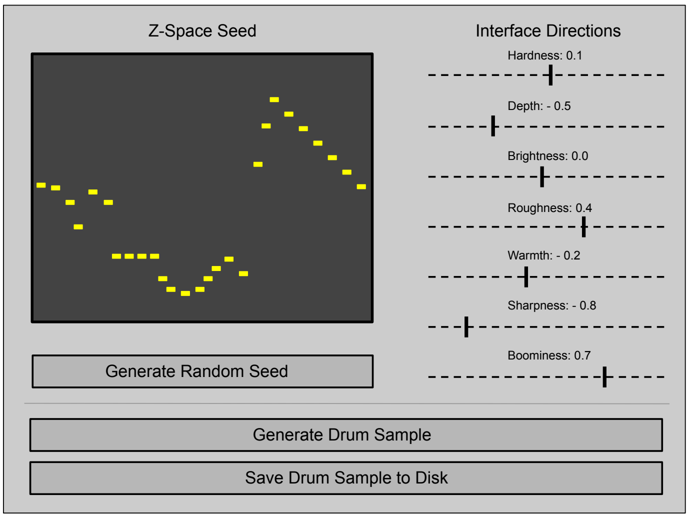
Figure 11.2: Graphical User Interface with SeFa directions.

InterfaceGAN. Similarly to the best-performing approach in the original SeFa paper (Shen & Zhou, 2021), we use the latent semantic factorization algorithm on the W space. These parameters are exposed as 7 horizontal faders, as shown in Figure 11.2, and are labeled and displayed in decreasing order of importance.

The last interface to evaluate is the supervised directions produced by InterfaceGAN, shown in Figure 11.3. InterfaceGAN requires the generation of examples from the trained network, as well as posterior manual or automatic annotation. To this end, we generate 10000 percussion sounds from our network and annotate them with the descriptors computed from AudioCommons timbral models (Pearce et al., 2017). This set of 7 descriptors were obtained from analyzing recurrent query terms related to timbral characteristics used for searching Freesound (Font et al., 2013). These features are hardness, depth, brightness, roughness, boominess, warmth and sharpness, and are calculated through regression models implemented in the AudioCommmons Extractor[76]. These exact descriptors have been previously used as conditioning features for controlling drum synthesis in previous work (Ramires et al., 2020b; Nistal et al., 2021).

To obtain the desired directions in InterfaceGAN, we use the latent embeddings in the W space, as these show higher classification accuracy in the SVM training stage (Shen et al., 2022). For the SVM training, we used 280 training

[76]https://github.com/AudioCommons/ac-audio-extractor



Hardness: 0.1
Depth: - 0.5
Brightness: 0.0
Roughness: 0.4
Warmth: - 0.2
Sharpness: - 0.8
Boominess: 0.7

Figure 11.3: Graphical User Interface with InterfaceGAN directions.

examples, 120 for validation, and the test set comprised the remaining 9600 samples. With this amount of data, the supervised algorithm was able to achieve a separation boundary which was able to separate negative and positive elements with decent accuracy as shown in Table 11.2. The parameters for the training, validation and test split were the ones used in the original InterfaceGAN article (Shen et al., 2022). The high validation accuracy, accompanied with lower values for test accuracy might indicate that this split might not be the best, as there are a lot of test examples and the training data is fairly limited.

We provide examples of manipulating each of the 7 features from SeFa and InterfaceGAN in the accompanying website[75].

11.4.5 Evaluation

Ultimately, evaluating user control over the generation of drum sounds focuses on the extent to which a user can express creativity. While designing a user study for evaluating the different approaches to parameterizing the StyleGAN2-based drum synthesizer, we determined that the Creativity Support Index (CSI) is the most relevant tool. The CSI is a psychometric survey designed to evaluate the extent to which a 'creative support system' can assist a user engaged in creative work – in this case synthesizing drums. The CSI measures six dimensions of creativity support: Exploration, Expressiveness, Immersion, Enjoyment, Results Worth Effort, and Collaboration. It allows researchers to evaluate how well a tool supports creative work overall and can pinpoint weaknesses in the various dimensions listed above. Subsequently, we present some example statements with which the test subjects are asked to rate from ‘Highly Agree’ (10) to ‘Highly Disagree’ (0), while they evaluate each of the three approaches to parameterization explored in the study: directly manipulating Z-Space, SeFa latent directions, and InterfaceGAN latent directions. Some example questions in the CSI are:

- Exploration: *“It was easy for me to explore many different ideas, options, designs, or outcomes, using this system or tool.”*
- Immersion: *“My attention was fully tuned to the activity, and I forgot about the system or tool that I was using.”*
- Results Worth Effort: *“What I was able to produce was worth the effort I had to exert to produce it.”*

As a final step in the evaluation, test subjects are asked to complete a ‘paired-dimension comparison’, which assesses how each subject values (or weights) each of the dimensions of creativity support already evaluated in the rating scale section. With these weights, the CSI score is determined by:

$$\begin{aligned}
&CSI_{score} = \\
&\quad(\text{CollaborationRating} \times \text{CollaborationWeigh} + \\
&\quad\text{EnjoymentRating} \times \text{EnjoymentWeight} + \\
&\quad\text{ExplorationRating} \times \text{ExplorationWeight} + \\
&\quad\text{ExpressivenessRating} \times \text{ExpressivenessWeight} + \\
&\quad\text{ImmersionRating} \times \text{ImmersionWeight} + \\
&\quad\text{ResultsWorthEffortRating} \times \\
&\qquad\qquad\text{ResultsWorthEffortWeight}) \;/\; 3.0
\end{aligned}$$

11.5 Results and Discussion

The evaluation was completed by 14 participants with various degrees of music production knowledge, from no music experience to professional music producers.

The results for the CSI evaluation are presented in Table 11.3. SeFa was the preferred interface by the participants, followed by the Z-space and the InterfaceGAN. SeFa has a clear preference with a margin of 4.14 in relation to the second best performing method. The results for InterfaceGAN and Z-Space are fairly similar, with a difference between the two of just 0.67. The low preference for InterfaceGAN could also be due to the low test-score obtained when finding the directions. However, when exploring this parameter space, the directions seemed to correspond to the desired attributes. Generally, participants reported having fun and were positively surprised by the ease of generating percussion sounds with each of the three techniques. Participants also commented on having enjoyed the exploratory process.

Z-Space	**SeFa**	**InterfaceGAN**
62.85 ± 11.91	$\mathbf{66.99 \pm 11.38}$	62.18 ± 11.40

Table 11.3: CSI scores for the 3 latent space navigation schemes under test.

Given the limited number of participants and their diverse backgrounds, the CSI scores unfortunately bear large confidence intervals and, therefore, these results cannot be said to be statistically significant. However, the trend towards favoring SeFa parameterization in this exploratory study was further echoed in anecdotes from participants during the debrief.

By interacting with the SeFa latent directions, it was reported that they were clearly related to specific concepts in the data space. While the first two parameters controlled the drum class and the amount of noise content respectively, the following controls controlled finer characteristics such as the decay time, depth, and boominess. The last controls labeled 5 and 6 did not impart any consistent variation in sounds generated across different latent samples.

Furthermore, participants reported an interesting user experience while interacting with the SeFa controls: If they wished to create kick drums, they could simply tweak the SeFa sliders (likely the first two sliders influencing drum class characteristics) to produce a kick sample for the currently chosen latent vector in the Z-Space. Then, subsequent randomly seeded latent vectors generated kick drums with different timbral characteristics. The same was found for hi-hats, toms, and snares, but less easily for other percussion types. This is likely attributed to the former having the highest representation in the training dataset.

On the other hand, a few participants characterized some InterfaceGAN parameters as redundant and not orthogonal to each other. Some participants reported issues regarding a lack of consistency from one seed to the next and not understanding the semantic concepts behind the parameters. Although having labeled directions could be desired for some experienced participants, testers valued the potential for exploring new timbres using SeFa without the need for music production knowledge – for example, the terminology employed by the InterfaceGAN UI.

In Table 11.4, we present the accumulated participant weights resulting from the ‘paired-dimension comparison’ in the CSI study.

CSI Weight	**Value**
Collaboration	0.64
Enjoyment	2.43
Exploration	**3.64**
Expressiveness	**3.36**
Immersion	2.00
Results Worth Effort	**3.21**

Table 11.4: CSI weights for each dimension.

From these results, it can be seen that the participants mentioned Exploration, Expressiveness and Results Worth Effort as the most important dimensions of creativity support for generating drum sounds. The high importance for Exploration could be the reason as to why SeFa scores highly in the CSI scale, as this system allows a controllable but serendipitous exploration of the latent space. Enjoyment and Immersion were still important but not as significant as the previously mentioned ones. Collaboration was the least significant dimension with a weight of almost 0.

11.6 Conclusions

In this work, we evaluated three methodologies for designing and editing drum sounds using GANs. To this end, a StyleGAN2 network was adapted to work with audio data and trained on a large private collection of drum sounds. We adapted two methodologies that showed promising results in controlling the generation of images – SeFa and InterfaceGAN – to our use case. We compared these approaches against the unconstrained navigation of the latent space of the network through a user test based on the Creativity Support Index. Our user study found that the unsupervised approach SeFa performed better for creative engagement with the StyleGAN2 network and we described the advantages and disadvantages of each interface.

Avenues for future work include the research and development of characteristics and classifiers better suited for the task of drum synthesis, to further improve the supervised approach InterfaceGAN. Redoing the experiment in a more specific scenario (e.g. replicating a drum sound or exploring drum sounds to fit a composition) could lead to a more confident result. Furthermore, creative possibilities of the StyleGAN2 network such as style mixing and adjusting magnitudes of noise at each resolution layer of the network could be included in the latent direction analyses explored in this chapter, to further enrich the quality of the resulting parameterization.

CHAPTER 12

Summary and Future Perspectives

In this chapter, we present the conclusions of this thesis. We start by summarizing the work we have conducted throughout this thesis and the main takeaways that have been presented at the end of each chapter. We will then present an overview of our most significant contributions that we have achieved throughout our research. Finally, we present our views on issues, limitations and future avenues for research in the our work and the topics covered by this thesis.

12.1 Summary

In this thesis, we explore ways in which deep learning can assist electronic music creation by providing new ways of browsing and exploring collections of music production sounds. To this end, we experimented with classification and generation techniques, the former to characterize musical material and the latter to create the sounds *in-between* the sounds present in music-making collections. With the large increase of sound material for music making that is available through online platforms, navigating both personal and online sound collections for music making can get extremely cumbersome. Previous studies conducted with artists have shown that this navigation is one of their main sources of disruption in the creative process. They ask for techniques to browse sounds based on their characterization. Despite the advances in recent technologies, the way of navigating collections of these production-ready sounds is still based on hierarchical tree directories and textual queries.

However, these large amounts of data which complicate navigation are what deep learning algorithms require to be trained. While automatic characterization had previously been researched with small datasets, handcrafted descriptors and traditional machine learning algorithms, there is now an opportunity to investigate it with deep learning methodologies further. Likewise, the possibilities offered by generative deep learning algorithms allow synthesizing sounds through high-level controls and without the need for extensive sound design knowledge.

In order to characterize music-making samples, we started by training a CNN architecture with a large-scale collection of instrumental sounds generated from digital synthesizers. The aim is to automatically classify the instrument in a one-shot sample into a category such as guitar, mallet or keyboard, in a similar way to the ones in commercial collections. We further investigated how a musically motivated data augmentation technique based on audio effects influenced classification accuracy and how sounds processed with these effects are harder to classify.

Following this work, we focused on the classification of loops. The biggest complication of this task was the lack of a high-quality dataset which had sufficient loops to power deep learning algorithms and which was free and available to use for research purposes. To face this difficulty, we created the FSLD dataset with Creative Commons loops from Freesound which comprises ≈ 10000 loops of which ≈ 3000 are annotated. An annotation tool was created to ease the annotation process and extend the dataset by the community.

With the FSLD, we were able to explore a new characterization task entitled automatic instrumentation role classification. The objective is to classify loops based on the roles they can take in a finished music composition: percussion, bass, chords, melody or effects. We compare several CNN architectures with different filters and pooling operators to evaluate their performance on this task. The augmentation technique from our work on one-shot instrument classification is combined with a music-motivated mixing of loops to create a larger and more balanced dataset. A use case for automatically identifying the structure of music pieces is also presented.

As a way of creating new samples from a collection, we then focused on generative deep learning. We took upon the use case of generative modeling of drum sounds based on high-level semantic characteristics. We used the Wave-U-Net feed-forward architecture conditioned on the energy envelope and seven timbral descriptors with semantic meaning (e.g. boominess, sharpness and roughness). We trained the model with a dataset of percussion one-shot sounds from Freesound that we curated and made available to the community.

The promising results we obtained from this first experiment led us to extend the architecture to generate full percussion loops. To this end, we collected drum loops from a public database and extended the feature set for conditioning. We extracted the same seven timbral features from the low, mid and high-frequency ranges and used the onset detection function provided by an automatic drum transcription algorithm. This allowed us to generate high-quality percussion loops, which can be controlled through a MIDI-like representation.

Finally, within the secondment in Native Instruments and to improve the synthesis quality of the one-shot generative model, we developed and trained a GAN-based architecture to generate one-shot drum sounds from NI's collection without any conditioning. We wanted to use this experiment to evaluate different parameters for controlling the percussion synthesis. Therefore, we compared the learned network embeddings, a dimensionality-reduced version of the embeddings and the previously used timbral features.

In the following section, we will summarize our contributions. Then, we will present the publications derived from this thesis and the code and datasets made publicly available. To conclude, we will detail the limitations of our work and present some future avenues for research derived from this thesis.

12.2 Summary of Contributions

The goal of this thesis is to enable new browsing possibilities for sounds in large-scale electronic music production collections. We focused on two main objectives: characterization and generation of EMP sounds. We will start by presenting our contributions and key results from Part I, which relates to sound classification, followed by the ones present in Part II, which relate to sound generation. The main contributions of this thesis can be summarized as follows:

- A new algorithm for instrument classification in one-shot sounds using CNNs (Chapter 4). A new data augmentation technique is proposed which is based on audio effects, improving classification accuracy. The NSynth dataset is used, which had not been used before for classification, and a baseline for further research using this dataset is provided. Finally, it is identified that using audio effects creates a severe problem when performing instrument classification.

- A new dataset of music loops for electronic music making (FSLD), and the annotation tool used for creating the dataset (Chapter 5). This dataset is the first public large-scale dataset of loops with high-quality annotations and with licenses which allow it to be used for scientific research and easily redistributed. The high amount of loops also enables it to be used for supervised deep-learning tasks. A detailed dataset analysis is presented, together with several use cases for tempo and key analysis, music generation and loop separation.

- Introduction of a new research task, Automatic Instrumentation Role Classification (AIRC), which can be evaluated using FSLD. A detailed comparison of several CNN architectures to address this task is provided. The classification accuracy obtained is high, and the algorithm performance in finding the structure of music pieces is on par with previous methods, which are more resource intensive. A novel data augmentation methodology for FSLD is proposed, and it is shown that it increases the accuracy in the structure classification task.

- A method for generating percussion sounds using a feed-forward convolutional network based on the Wave-U-Net is presented (Chapter 9). As conditioning, handcrafted features designed by MIR experts are used, which relate to semantic timbral attributes which music makers understand. It is shown that the conditioning works properly and that changing the parameters indeed modifies the values of the features in the output. The use of the Wave-U-Net-based architecture leads to a faster generation. Also part of this contribution is a curated dataset of one-shot percussive sounds from Freesound, which is used to train the model.

- A method for generating longer sequences of audio with more than one instrument (Chapter 10). This is the first deep learning system which can generate drum loops and which can be controlled using semantic high-level parameters and using a similar representation to MIDI. Various loss functions are tested and the synthesis quality of each one is compared. The architecture used allows extremely fast generation of loops, even without the use of a GPU.

- A user-based evaluation using Creativity Support Index (CSI) of three sets of parameters used for controlling drum synthesis in GANs (Chapter 11). A GAN architecture is proposed based on StyleGAN2 to evaluate relevant directions on the latent embeddings which can be used for controlling the sound generation in a meaningful way. The sets of parameters are obtained using supervised and unsupervised algorithms.

- Two software libraries for i) the tonal description of audio signals; and for ii) interacting with Freesound from audio plug-ins (Appendix C and D).

12.3 Limitations and Future Work

In this section, we present the limitations we believe our work has, as well as some of the possible avenues for future research in these topics.

12.3.1 Classification of One-Shot Sounds

Our work on the automatic classification of one-shot sounds opened several avenues for future research on the topic. To begin with, the accuracy can be fairly improved as the one obtained by our models did not surpass 75%. To achieve more reliable predictions, one of the possible pathways would be to experiment with higher capacity models, which should be able to make better use of the high amount of data than the single-layer architecture we used.

Another possible way to improve these values is to extend the data augmentation pipeline to apply a more diverse set of effects with different settings. In our work, we used a fixed set of effects with their default parameters. To have a more reliable data augmentation and evaluation, more audio effects can be used and their parameters randomized, which will lead to a higher diversity in the train and test data. The higher variety in the training set can lead to a higher diversity of examples seen by the model at training which can lead to it learning techniques which make it more robust to the perturbations created. In the test set, it can lead to a more precise evaluation of how it performs on the transformed data.

Another possible avenue for future work would be adding acoustic and electric instruments to the training data, as the dataset we used only contains digital ones. This could lead to a trained model which can be extremely robust and fit to any instrument classification task.

12.3.2 Freesound Loop Dataset and Annotation Tool

The main limitations of the Freesound Loop Dataset are the number of loops, the amount which is annotated and the annotation detail. To increase the number of loops in FSLD, one possible way would be to collaborate with commercial sample providers to add their loops to the dataset. Integrating these loops would require modifying their original metadata to fit the taxonomy in FSLD. However, as most of the characteristics annotated were chosen considering commercial collections, this should not pose a hard task.

One way to increase the amount of annotated loops within FSLD would be to promote the use of the annotation tool we developed. Ideally, methods to do this would be finding ways to get the MIR community involved in annotating more loops or hiring experts to do this procedure.

A possible limitation of FSLD is the level of detail at which the annotations are. It would be interesting to have a finer grain of specificity in annotated characteristics. The biggest difference between this dataset and commercial collections is that we annotate the instrumentation role instead of the instrument. In the case of Freesound, as some of the sounds are highly experimental, it is hard to identify the instrument which created the sounds but not the instrumentation roles. However, having a more detailed instrument annotation would enable more possibilities in future work on this dataset.

Finally, future avenues for research using this dataset are related to the applications for which it can be used. Tempo, key and chord estimation specific to loops are tasks that can be further advanced using the dataset. Also, automatic music creation and automatic generation of new loops based on the ones in the dataset are interesting possibilities for future work.

12.3.3 Automatic Instrumentation Role Classification

Our work in automatic instrumentation role classification opened new paths for research on both improving the classification accuracy as well as developing new applications based on the algorithm. Although the accuracy obtained in our AIRC algorithms was quite high, there is always room for improvement with more data and higher capacity models.

When applying the models trained on loops to full music pieces, one thing we saw that could be improved was the resolution of the results, as the results currently being provided are for 4-bar segments. Attention-based techniques or training with smaller fragments of data can lead to this finer resolution, as well as more explainability of what parts of the audio are being taken into consideration for the final output.

Regarding applications, we envision several use cases for which the algorithm can be used. The first one is using the predictions on musical pieces as visual cues for DJs to understand a track's structure. This information can also be used for automatic DJing systems to create smoother transitions. Knowing the structure of a music piece also allows copying its structure into an unfinished piece. If a music maker only has, for example, the chorus of a track but does not know how to structure it, they can copy the output of our system and apply it. Finally, one very interesting avenue of research would be finding samples from existing music. This would correspond to a faster version of *crate-digging*, a practice common among hip-hop producers where they go through crates of records to find isolated instrumental sections that can be used in their compositions. Our system can identify parts composed of only percussive elements or melodic/harmonic elements, which, in vinyl records, was only possible through hearing the full composition.

12.3.4 Generation of Percussion Sounds

In this subsection, we will present the main drawbacks we faced and avenues for future work in the generation of percussive one-shots and loops. These will be presented together for the three chapters related to this topic because the issues and promising directions are fairly similar. We believe the major drawbacks faced are the synthesized audio quality of the generated examples, the evaluation methodologies, the parameters we used for controlling the synthesis and the large amount of data required to train these models.

The main issue in our work on using the Wave-U-Net for generating drum sounds was the insufficient audio quality of the generated sounds to be used for professional audio applications. This was probably due to the choice of loss functions, as we have seen in the loop generation work. Although the synthesis quality improved significantly from one work to another, we find that there is still room for improvement, specifically in developing new loss functions specifically targeted at audio and the task of percussion synthesis. Furthermore, we believe better evaluation metrics to monitor the achieved audio quality during the training are required, as we will further develop in the following paragraph. When using the GAN architecture, we saw that synthesis quality increased even further. One of the main advantages of this architecture is the use of its specific loss function which is based on the discriminator accuracy, and not directly calculated from the audio. Novel architectures with special training paradigms can further help increase audio quality.

Evaluating the audio quality of deep learning generated sounds is normally done through user tests which compare the generated sounds to a reference or through a reference-free metric such as the Fréchet Audio Distance (FAD). The main concern with user testing is the time and effort which is needed to do it in an appropriate manner and possibly associated financial costs. Furthermore, these tests cannot assign an absolute value for the audio quality of the generated sound, as this is a highly subjective aspect of sound. The use of a reference-free metric such as FAD can handle all these shortcomings and is shown to be able to quantify the amount of distortion present in processed audio. However, this metric relies on statistics extracted from both the training set and the generated data and, in the case the generated data does not fit a similar distribution to the training data (e.g. the model creates more snares than kick drums, when kick drums are more prominent in the training data), the values will be less accurate. Therefore, there is a need for better metrics to quantify audio quality in generative deep learning models, which do not require user testing and do not relate to the distribution of the generated audio.

Another difficulty in evaluating generative models is assessing their usability and how they enable creativity. Evaluating these aspects is also done through user studies and interviews, which can be cumbersome. In the work we conducted, we used the AudioCommons timbral features as parameters for the generation as we believe they are semantically rich and music makers can understand them. However, as some users mentioned in our final drum synthesis work, they found that these parameters were not decoupled from each other. Obviously, this could be due to the supervised method for finding latent directions not working as well as it should, and this is, consequently, a possible improvement. However, the selection of the best parameters for controlled drum synthesis is still an open issue which needs significant work. Interviews with music makers and professional sound designers should be done, and an iterative procedure of testing with users and improving these systems would be ideal. Furthermore, for these algorithms to reach the end-user, appropriate user interfaces and integration with commonly used systems still have to be done.

Finally, to adapt our models to the collections of music makers and allow them to train them on their collections, new techniques which require fewer data have to be researched. This would enable music makers to have a trained model which generates data similar to the ones they commonly use and should therefore prefer. This would also enable a new type of collection to be shared between artists, where, instead of a sample pack, the artist could share their own drum synthesizer.

Appendix A

Glossary

A.1 Acronyms

AdaIN	Adaptive Instance Normalisation
AE	Autoencoder
AI	Artificial Inteligence
AIRC	Automatic Instrumentation Role Classification
API	Application Programming Interface
ASDS	Adversarial Synthesis of Drum Sounds
BCE	Binary Cross Entropy
BPM	Beats Per Minute
CD	Compact Disk
CNN	Convolutional Neural Network
CREM	Centre de Recherche en Ethnomusicologie
CSI	Creativity Support Index
DAW	Digital Audio Workstation
DFT	Discrete Fourier Transform
DS	Drum Synthesis
EDM	Electronic Dance Music
ELU	Exponential Linear Unit
EM	Electronic Music
EMP	Electronic Music Production
FAD	Fréchet Audio Distance
FFT	Fast Fourier Transform
FSLD	Freesound Loop Dataset
FSOSPS	Freesound One-Shot Percussive Sounds
GAN	Generative Adversarial Network
GMP	Global Max Pooling
GPU	Graphical Processing Unit

GRM	Groupe de Recherches Musicales
HPCP	Harmonic Pitch Class Profiles
IIR	Infinite Impulse Response
IRAM	Instrumentation Role Activation Map
IRMAS	Instrument Recognition in Musical Audio Signals
ISMIR	International Society for Music Information Retrieval
KID	Kernel Inception Distance
MA	Macro Average
MFCC	Mel-Frequency Cepstral Coefficients
MIDI	Musical Instrument Digital Interface
MIP-Frontiers	New Frontiers in Music Information Processing
MIR	Music Information Retrieval
MIREX	Music Information Retrieval Evaluation eXchange
NI	Native Instruments GmbH
NMFD	Non-Negative Matrix Factorization Deconvolution
NTF	Non-Negative Tensor Factorization
PCA	Principal Component Analysis
PR-AUC	Area Under the Precision-Recall Curve
ReLU	Rectified Linear Units
RNN	Recurrent Neural Network
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
RTF	Radiodiffusion Télévision Françaises
RWC	Real World Computing
SeFa	Semantic Factorization
SGD	Stochastic Gradient Descent
STFT	Short-Time Fourier Transform
SVM	Support-Vector Machine
TIV	Tonal Interval Vectors
VAE	Variational Autoencoder
VST	Virtual Studio Technology

Appendix B

Publications, Open Research and Reproducibility

Open science is one of the main focuses within the MIP-Frontiers project, and we have tried to follow its principles as much as possible. We have made most of our research output available publicly to everyone, for free and under copyleft licenses. Open science helps the research community firstly by promoting more transparent results which other researchers can verify. Sharing the code and data of experiments can also increase their impact in academia and industry by enabling easier reproducibility and implementation of the research.

B.1 Publications by the Author

The work we conducted throughout this thesis was published in multiple conferences with high impact in the MIR and music technology field. These conferences were also selected due to publishing their proceedings freely and in an open manner. Additionally, to have all the papers available through the same place for free access and so that they are easier to disseminate, we published the preprint versions on arxiv[77] and e-Repositori[78] under open licenses. The publications that were conducted during the PhD are the following:

B.1.1 Articles in Peer-Reviewed Conferences

- **Ramires, A.**, & Serra, X. (2019). Data augmentation for instrument classification robust to audio effects. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

[77]https://arxiv.org
[78]https://repositori.upf.edu

- **Ramires, A.**, Font, F., Bogdanov, D., Smith, J., Yang, Y.H., Ching, J., Chen, B.Y., Wu, Y.K., Wei-Han, H., & Serra, X. (2020). The Freesound Loop Dataset and Annotation Tool. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR).*

- Ching, J., **Ramires, A.**, & Yang, Y.H. (2020). Instrument Role Classification: Auto-tagging for Loop Based Music. In *Proceedings of The 2020 Joint Conference on AI Music Creativity (MuMe + CSMC).*

- Drysdale, J., **Ramires, A.**, Serra, X., & Hockman, J. (2022). Improved Automatic Instrumentation Role Classification and Loop Activation Transcription. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

- **Ramires, A.**, Chandna, P., Favory, X., Gómez, E., & Serra, X. (2020). Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 786–790).

- Chandna, P., **Ramires, A.**, Serra, X., & Gómez, E. (2021). LoopNet: Musical Loop Synthesis Conditioned on Intuitive Musical Parameters. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 3395-3399).

- **Ramires, A.**, Juras, J., D. Parker, J., & Serra, X. (2022). A Study of Control Methods for Percussive Sound Synthesis Based on GANs. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

- **Ramires, A.**, Bernardes, G., Davies, M., & Serra, X. (2020). TIV.LIB: An Open-Source Library for the Tonal Description of Musical Audio. In *Proceedings of the International Conference on Digital Audio Effects (DAFx).*

B.1.2 Extended Abstracts

- **Ramires, A.**, Font, F., & Serra, X. (2019). Freesound JUCE API. In *Audio Developer Conference 2019 (ADC19).*

B.1.3 Perfomances

- Roma, G., Font, F., & **Ramires, A.** (2021). Floop Jam. In *Proceedings of the International Web Audio Conference.*

B.1.4 Supervision of Master Thesis

- Pérez Fernández, M. (2020). Harmonic compatibility for loops in electronic music. (MSc Thesis, Universitat Pompeu Fabra).

B.2 Open Source Software

Making research reproducible and openly accessible enables faster scientific progress and makes it easier for companies to deploy it. The use of open licenses in our code allows it to be deployed in professional tools as long as these use similar licenses. Furthermore, providing the code to the public can enable further research in the fields covered by the thesis by providing an easy way of reproducing our results and experimenting with novel techniques built upon our experiments. Despite the extra time required to make our experiments easily reproducible, we believe the advantages outweigh these costs. To enable the general audience to experiment with our drum generation tools, we have created interactive demos on Google Colaboratory for the research in Chapters 9 and 10.

The code for reproducing our experiments is available at:

- **Chapter 4**: https://github.com/aframires/instrument-classifier
- **Chapter 5**: https://github.com/aframires/freesound-loop-annotator
- **Chapter 6**: https://github.com/aframires/airc
- **Chapter 9**: https://github.com/pc2752/percussive_synth
- **Chapter 10**: https://github.com/aframires/drum-loop-synthesis
- **Chapter 11**: https://github.com/aframires/stylegan2-ada-pytorch
- **Appendix C**: https://github.com/aframires/TIVlib
- **Appendix D**: https://github.com/mtg/freesound-juce

B.3 Datasets

On the field of MIR applied to EMP, one of the main concerns is the lack of datasets which can be used in research and shared with the public. This is especially important when working with deep learning models that require big amounts of data to be able to generalize well to unseen examples. Leveraging Freesound, we were able to create two large-scale datasets:

- The Freesound Loop Dataset (FSLD) is available at https://doi.org/10.5281/zenodo.3967852 and contains 9,455 loops from Freesound.org and the corresponding annotations. All the loops have annotations on the tempo and genre provided by the users who uploaded them, an automatically obtained key label and the textual metadata from Freesound. Approximately 3 000 of these loops also have tempo, key, genre and instrumentation annotations made by the experts who contributed. This dataset was already downloaded 4 000 times.

- The Freesound One-Shot Percussive Sounds (FSOSPS) dataset is available at https://doi.org/10.5281/zenodo.3665275 and contains the 10254 one-shot (single event) percussive sounds from Freesound.org and the corresponding timbral analysis used in Ramires et al. (2020b). These sounds were obtained by querying Freesound with words associated with percussive instruments, such as "percussion", "kick", "wood" or "clave". We manually selected sounds which only had one event and were not loops and standardized their format in terms of length, normalization and sampling rate. This dataset was downloaded 490 times.

Appendix C

TIV.lib: an open-source library for the tonal description of musical audio

In this appendix, we present TIV.lib, an open-source library for the content-based tonal description of musical audio signals. Its main novelty relies on the perceptually-inspired Tonal Interval Vector space based on the Discrete Fourier transform, from which multiple instantaneous and global representations, descriptors and metrics are computed—e.g., harmonic change, dissonance, diatonicity, and musical key. The library is cross-platform, implemented in Python and the graphical programming language Pure Data, and can be used in both online and offline scenarios. Of note is its potential for enhanced Music Information Retrieval, where tonal descriptors sit at the core of numerous methods and applications.

C.1 Introduction

In Music Information Retrieval (MIR), several libraries for musical content-based audio analysis, such as Essentia (Bogdanov et al., 2013), Librosa (McFee et al., 2015b), and madmom (Böck et al., 2016) have been developed. These libraries have been widely adopted across academia and industry as they promote the fast prototyping of experimental methods and applications ranging from large-scale applications such as audio fingerprinting and music recommendation, to task-specific MIR analysis including chord recognition, structural segmentation, and beat tracking.

The tonal domain of content-based audio descriptors denotes all attributes related to the vertical (i.e., harmonic) and horizontal (i.e., melodic and voice-leading) combination of tones, as well as their higher-level governing principles, such as the concept of musical key. The earliest research in this domain was driven by the methods applied to symbolic representations of music, e.g., MIDI files. The jump from symbolic to musical audio domain raises significant problems and requires dedicated methods, as polyphonic audio-to-symbolic transcription remains a challenging task (Benetos et al., 2018). While the state of the art (Curtis Hawthorne et al., 2018; Ycart & Benetos, 2018) in polyphonic music transcription has advanced greatly due to the use of deep neural networks, it remains largely restricted to piano-only recordings.

One of the most prominent tonal audio descriptors is the chroma vector. This representation divides the energy of the spectrum of an audio signal in the 12 tones of the western chromatic scale across all octaves. This leads to a 12-element vector where each element corresponds to the energy of each pitch class. Throughout this work, this vector will be referred to as the pitch profile. Many algorithms for this representation have been proposed, including Pitch Class Profiles (Fujishima, 1999), Harmonic Pitch Class Profiles (HPCP) (Gómez, 2006b), the CRP chroma (Muller & Ewert, 2010), and the NNLS chroma (Mauch & Dixon, 2010). Stemming from this 12-element vector, many metrics and systems have been proposed, for key detection, chord recognition, cover song identification, mood recognition, and harmonic mixing. Yet, despite their fundamental role in many MIR tasks, tonal descriptors are not only less prominent in existing content-based audio libraries, in comparison with rhythmic or timbral descriptors (Bogdanov et al., 2013), but also their perceptual basis is of limited scope (Bernardes et al., 2017b).

In the context of the aforementioned limitations, we present the TIV.lib, a cross-platform library for Python and Pure Data, which automatically extracts multiple perceptually-aware tonal descriptions from polyphonic audio signals, without requiring any audio-to-symbolic transcription stage. It owes its conceptual basis to ongoing work within music theory on the Discrete Fourier Transform (DFT) of pitch profiles, which has been extended to the audio domain (Bernardes et al., 2016a). The hierarchical nature of the Tonal Interval Vectors (TIV) space allows the computation of instantaneous and global tonal descriptors including harmonic change, (intervallic) dissonance, diatonicity, chromaticity, and key, as well as the use of distance metrics to extrapolate different harmonic qualities across tonal hierarchies. Furthermore, it can enable the efficient retrieval of isolated qualities or those resulting from audio mixes in large annotated datasets as a simple nearest neighbour-search problem.

The remainder of this appendix is organized as follows. Section C.2 provides an overview of the ongoing work on pitch profiles Discrete Fourier Transform (DFT)-based methods within music theory, followed by a description of the recently proposed Tonal Interval Vectors (TIV) space, which extends the method to the audio domain. Section C.3 provides a global perspective of the newly proposed TIV.lib architecture. Section C.4 details the mathematical and musical interpretation of the description featured in TIV.lib and, finally, Section C.5 discusses the scope of application scenarios of the library and Section C.6 provides perspectives on future work.

C.2 Related work

C.2.1 Tonal pitch spaces

Within the research literature, numerous tonal pitch spaces and pitch distance metrics have been proposed (Shepard, 1962; Lerdahl, 2004; Tymoczko, 2010; Chew, 2007). They aim to capture perceptual musical phenomena by geometrical and algebraic representations, which quantify and (visually) represent pitch proximity. These spaces process pitch as symbolic manifestations, thus capturing musical phenomena under very controlled conditions, with some of the most prominent spaces discarding the pitch height dimension by collapsing all octaves into a 12-tone pitch space.

Attempts to represent musical audio in the aforementioned spaces have been pursued (Chuan & Chew, 2005; De Haas et al., 2008) by adopting an audio-to-symbolic transcription stage. Yet, polyphonic transcription from musical audio remains a challenging task which is prone to error.

Recently, Bernardes et al. (2016a) proposed a tonal pitch space which maps chroma vectors derived from audio signals driven into a perceptually inspired Discrete Fourier Transform (DFT) space. It expands the aforementioned pitch spaces with strategies to process the timbral/spectral information from musical audio.

C.2.2 From the DFT of symbolic pitch distributions to the Tonal Interval Vector space

In music theory, the work proposed by Quinn (2007) and followed by Amiot (2016); Yust (2019); Tymoczko & Yust (2019) on the DFT of pitch profiles, has been shown to elicit many properties with music-theoretic value. Moreover, in Dawson et al. (2020) DFT-based pitch spaces were shown to capture human perceptual principles.

In the Fourier space, a 6-element complex vector, corresponding to the $1 \leq k \leq 6$ DFT coefficients, is typically adopted. The magnitude of the Fourier coefficients has been used to study the shape of pitch profiles, notably concerning the distribution of their interval content. This allows, for example, to quantify diatonic or chromatic structure (see Section C.4 for a comprehensive review of the interpretations of the coefficients). The phase of the pitch profiles in the Fourier space reveals aspects of tonal music in terms of voice-leading (Tymoczko, 2008), tonal regions modelling and relations (Yust, 2017), and the study of tuning systems (Amiot, 2016). In summary, the magnitude of the pitch profiles express harmonic quality and the phases harmonic proximity.

Recently, a perceptually-inspired equal-tempered, enharmonic, DFT-based TIV space (Bernardes et al., 2016a) was proposed. One novelty introduced by this newly proposed space in relation to remaining Fourier spaces was the combined use of the six coefficients in a TIV, $T(k)$. Moreover, the perceptual basis of the space is guaranteed by weighting each coefficients by empirical ratings of dyad consonance, $w_a(k)$. $T(k)$ allows the representation of hierarchical or multi-level pitch due to the imposed L_1 norm, such that:

$$\begin{gathered} T(k) = w_a(k) \sum_{n=0}^{N-1} \bar{c}(n) e^{\frac{-j2\pi kn}{N}}, \\ k \in \mathbb{Z} \quad \text{with} \quad \bar{c}(n) = \frac{c(n)}{\sum_{n=0}^{N-1} c(n)} \end{gathered} \tag{C.1}$$

where N=12 is the dimension of the chroma vector, $c(n)$, and k is set to $1 \leq k \leq 6$ for $T(k)$ since the remaining coefficients are symmetric. The weights, $w_a(k) = \{3, 8, 11.5, 15, 14.5, 7.5\}$, adjust the contribution of each dimension k of the space to comply with empirical ratings of dyad consonance as summarised in Huron (1994). $w_a(k)$ accounts for the harmonic structure of musical audio driven from an average spectrum of orchestral instruments (Bernardes et al., 2017b).

C.3 TIV.lib: Implementation

The TIV.lib includes several signal-processing functions or descriptors for characterising the tonal content of musical audio. This library is implemented in Python, using only Numpy and Scipy as dependencies and Pure Data, with both available to download at: http://bit.ly/2pBYhqZ. Illustrative analysis of musical audio examples for the descriptors are provided in the Python download link as a Jupyter Notebook. The Python implementation targets batch offline processing and the Pure Data implementation online processing.

As an input, the library takes 12-element chroma vectors, $c(n)$, from which TIVs, $T(k)$, are then computed. [79] Any input representation will have an effect on the space, as such we leave the choice of which chroma representation up to the user in order to best fit the problem at hand. Although the system is

[79] A tutorial example on the extraction of HPCP representations from audio is provided in the library package, both using Essentia and Librosa.

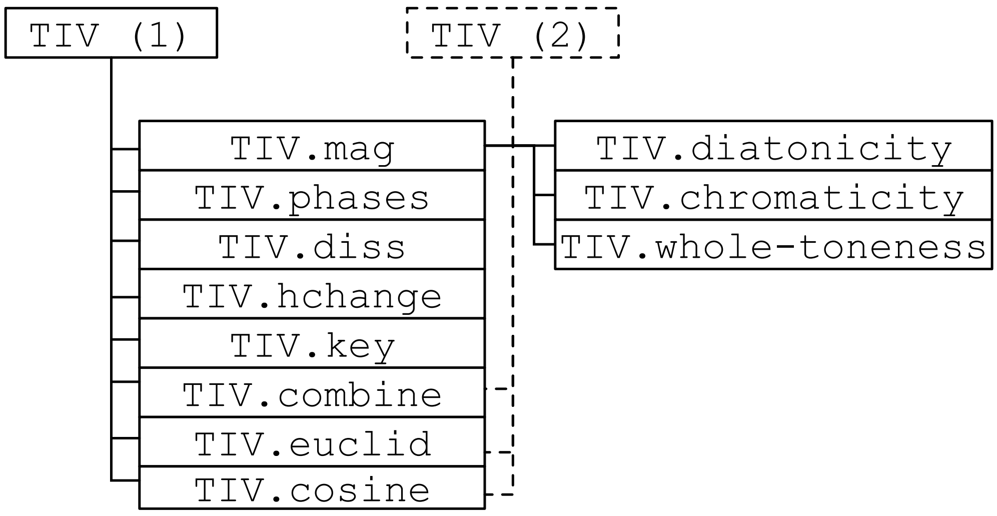


Figure C.1: A graph of the dependencies of the feature extraction modules of TIV.lib. The algorithms connected to TIV(2) through a dashed line require two inputs for the feature calculation.

agnostic to the chosen chroma, we recommend the "cleanest" chroma representation, i.e., that which is closest to a symbolic representation, to be selected. The time scale of the TIV is dependent of the adopted window size during the chroma vector computation. For instantaneous TIVs, a single-window chroma vector can be used as input. For global TIVs, consecutive chroma vectors can be averaged across the time axis prior to the TIV computation.

In Figure C.1 we present the architecture of TIV.lib. In this graph of dependencies we can see the algorithms that have been implemented and which classes they require for their calculation.

C.4 TIV.lib: Algorithms

This section details the functions included in the TIV.lib, focusing on their mathematical definition and musical interpretation.

TIV is a 6-element complex vector, which transforms chroma into an interval vector space by applying Eq. C.1, an L_1-norm weighted DFT. The resulting space dimensions combine intervallic information in the coefficients' magnitude and the tonal region (i.e., musical key area) it occupies in the coefficients' phase. The mapping between chroma and the TIV retains the bijective property of the DFT and allows the representation of any variable-density pitch profile in the chroma space as a unique location in the TIV space.

TIV.mag is a 6-element (real) vector that reports the magnitude of the TIV elements $1 \leq k \leq 6$, such that:

$$mag(k) = ||T(k)|| \tag{C.2}$$

It provides a characterisation of the harmonic quality of a pitch profile, namely its intervallic content, distilling the same information as the pitch-class interval vector (Forte, 1964, 1973). Mathematically, it is well-understood that a large magnitude in $T(k)$ coefficients indicates how evenly the pitch profile can be divided by N/k. Musically, the work on the DFT of pitch profiles (Bernardes et al., 2016a; Quinn, 2007) emphasizes the association between the magnitude of Fourier coefficients and tonal qualities: $||T(1)|| \leftrightarrow chromaticity$, $||T(2)|| \leftrightarrow dyadicity$, $||T(3)|| \leftrightarrow triadicity$, $||T(4)|| \leftrightarrow diminished\, quality$, $||T(5)|| \leftrightarrow diatonicity$, $||T(6)|| \leftrightarrow whole-toneness$. Please refer to Amiot (2016); Yust (2019) for a comprehensive discussion on the interpretation of the DFT coefficients.[80] One distinct property of the TIV.mag vector is its invariance under transposition or inversion (Amiot, 2016). For example, all major triads or harmonic minor scales share the same Fourier magnitude, hence the same TIV.mag vector [81].

TIV.phases is a 6-element (real) vector that reports the phases (or direction) of the TIV coefficients $1 < k < 6$, such that:

$$phases(k) = \angle T(k) \tag{C.3}$$

It indicates which of the transpositions of a pitch profile quality is under analysis (Hoffman, 2008), as transposition of a pitch profile by p semitones, i.e., circular rotations of the chroma, $c(n)$, rotates the $T(k)$ by $\varphi(p) = \frac{-2\pi kp}{N}$. TIV phases are also associated with regional (or key) areas, whose diatonic set is organised as clusters in the TIV space (Bernardes et al., 2016a; Yust, 2017).

TIV.combine computes the resulting TIV from mixing (or summing) multiple TIVs representing different musical audio signals. Due to the properties of the DFT space, this operation can be efficiently computed as a linear combination of any number of TIVs, $T(k)$. Given TIVs $T_1(k)$ and $T_2(k)$, their linear combination, weighted by their respective energy, a_1 and a_2, is given by:

$$T_{1+2}(k) = \frac{T_1(k) \cdot a_1 + T_2(k) \cdot a_2}{a_1 + a_2} \tag{C.4}$$

a_1 and a_2 are retrieved from the discarded DC components $T_1(0)$ and $T_2(0)$.

TIV.chromaticity reports the level of concentration of a sonority in a specific location of the chromatic pitch circle as a value within the [0,1] range, computed

[80] We note for each of these single Fourier coefficient quantities that the effects of the weights can be factored out.

[81] Note that the phases, discarded here, will differ. As such, the uniqueness property of the TIV is maintained as it combines both magnitude and phase information.

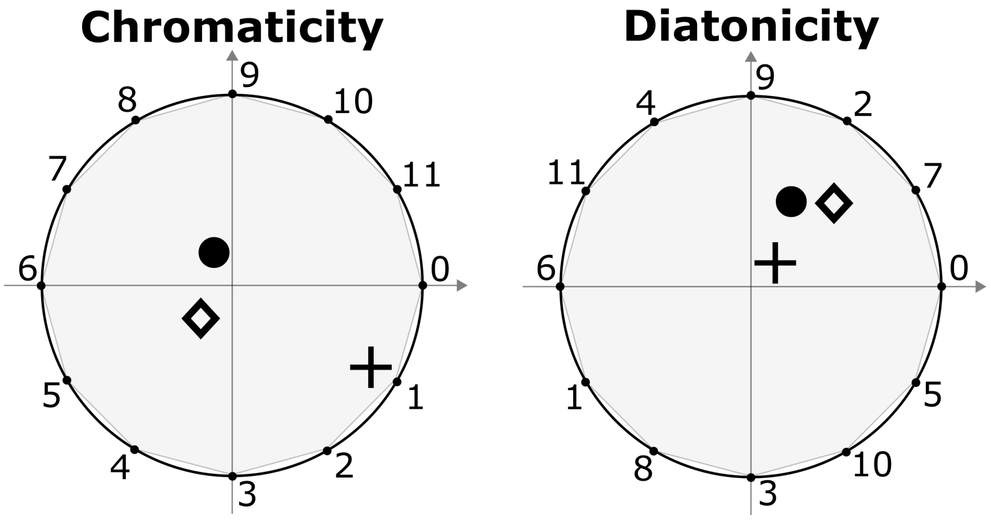


Figure C.2: Two DFT coefficients interpreted as chromaticity and diatonicy. Three TIV are plotted for comparison: C major chord {0,4,7} (◇), 3-note chromatic cluster {0,1,2}(+), and C major scale {0,2,4,5,7,9,11}(•)

as the magnitude of the $T(1)$ normalized to unity: $\frac{||T(1)||}{w_a(1)}$. This value is close to 0 for sounds exhibiting energy in evenly-spaced pitch classes (such as typically tonal chords and scales) and close to 1 for chromatic pitch aggregates.

TIV.diatonicity reports the level of concentration of a sonority within the circle of fifths as a value within the [0,1] range. The larger the magnitude of the $T(5)$ normalized to unity, $\frac{||T(5)||}{w_a(5)}$, the higher the level of diatonicity.

TIV.whole-toneness reports the proximity to one of the two existing whole-tone collection within the 12-tone equal temperament tuning. The level of whole-toneness is reported within the [0,1] range resulting from the magnitude of the $T(6)$ normalized to unity, such that: $\frac{||T(6)||}{w_a(6)}$.

Fig. C.2 shows the DFT coefficients from which we extract chromaticity and diatonicity descriptions as the magnitude of $T(1)$ and $T(5)$, respectively. We plot pitch profiles that aim to illustrate the behaviour of each coefficient in eliciting the chromatic and diatonic character of the C major chord and C major scale as well as chromatic 3-tone cluster by inspecting their magnitude. Note that the magnitude of both the C major chord and C major scale, two prototypical diatonic pitch profiles, clearly have greater magnitude in the diatonic $T(5)$ coefficient in comparison with the three-note cluster, a prototypical chromatic profile. Conversely, in the chromatic $T(1)$ coefficient, the magnitudes of the above pitch profiles show the expected opposite behaviour, thus mapping the three-note cluster further from the centre.

TIV.euclid and TIV.cosine compute the Euclidean, E, and cosine, C, distance between two given TIVs, $T_1(k)$ and $T_2(k)$, using Eqs. C.5 and C.6, respectively.

$$E\{T_1, T_2\} = \sqrt{||T_1 - T_2||^2} \tag{C.5}$$

$$C\{T_1, T_2\} = \frac{T_1 \cdot T_2}{||T_1|| ||T_2||} \tag{C.6}$$

The cosine distance (i.e., the angular distance) between TIVs can be used as an indicator of how well pitch profiles "fit" or mix together. For example, it quantifies the degree of tonal proximity of TIV mixtures, or informs which translation or transposition of a TIVs best aligns with a given key. Conversely, Euclidean distances between TIVs relate mostly to melodic (or horizontal) distance. It captures the neighbouring relations observed in the *Tonnetz*, where smaller distances agree with parsimonious movements between pitch profiles. Please refer to Tymoczko & Yust (2019); Tymoczko (2008) for a comprehensive discussion on this topic.

TIV.hchange computes a harmonic change detection function across the temporal dimension of an audio signal. Peaks in this function indicate transitions between regions that are harmonically stable. We compute a harmonic change measure, λ, for an audio frame m as the Euclidean distance between frames $m+1$ and $m-1$ (Eq. C.7), an approach inspired by Harte et al. (2006), which can be understood as adopting three coefficients out of the $1 \leq k \leq 6$ of the TIV, $T(k)$, i.e., those corresponding to the circle of fifths, the circle of minor thirds, and the circle of major thirds.

$$\lambda_m = \sqrt{||T_{m-1} - T_{m+1}||^2} \tag{C.7}$$

TIV.diss provides an indicator of (interval content) dissonance, as the normalized TIV magnitude subtracted from unity, $1 - \frac{|T(k)|}{|w_a(k)|}$. This perceptually-inspired indicator stems from the weighted magnitude of the TIV coefficients, which rank the intervals $1 \leq k \leq 6$ to match empirical ratings of dissonance within the Western tonal music context (Bernardes et al., 2017b, 2016a).

TIV.key infers the key from an audio signal as a pitch class (tonic) and a mode (major or minor). It is computed as the Euclidean distance from the 24 major and minor key TIVs, $T_r^{p\star}(k)$, defined as the shifts (i.e. rotation) of the 12 major and 12 minor profiles, p, by Temperley (Temperley, 1999) or Sha'ath (Sha'ath, 2011), such that:

$$R_{min} = \text{argmin}_r \sqrt{||T \cdot \alpha - T_r^{p\star}||^2} \tag{C.8}$$

where $T_r^{p\star}$ are 24 major and minor key profiles TIVs, p. When $r \leq 11$, we adopt the major profile and when $r \geq 12$, the minor profile. α is a bias introduced to balance the distance between major and minor keys. Optimal values

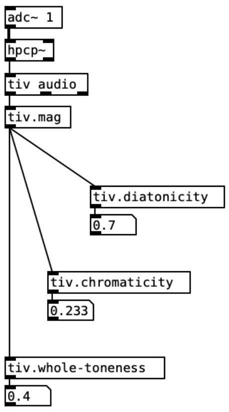


Figure C.3: Pure Data patch for online computation of diatonicity, chromaticity and whole-toneness harmonic qualities of a live input audio signal, using the TIV.lib.

of $\alpha = 0.2$ and $\alpha = 0.55$ have been proposed in Bernardes et al. (2017a) for the Temperley (Temperley, 1999) and Shat'ath (Sha'ath, 2011) key profiles, respectively. The output is an integer, R_{min}, ranging between $0-11$ for major keys and $12-23$ for minor keys, where 0 corresponds to C major, 1 to C# major, and so on through to 23 being B minor.

C.5 Applications and Perspectives

Following the emerging body of music theory literature on the DFT of pitch profiles and the continuous work of the TIV space, we implemented the perceptually-inspired TIV.lib in Python and Pure Data. The former aims at batch offline processing and the latter mostly at online or real-time processing, but allowing offline computations as well.

Figure C.3 shows an example usage of the TIV.lib functions for computing the diatonicity, chromaticity and whole-toneness harmonic qualities in Pure Data.

In order to achieve the same result in Python, the following code can be executed:

```
import TIVlib as tiv

ex_tiv = tiv.TIV.from_pcp(example_chroma)
ex_wholetoneness = ex_tiv.wholetoneness()
ex_diatonicity = ex_tiv.diatonicity()
ex_chromacity = ex_tiv.chromaticity()
```

We now provide an example usage of this library for extracting tonal features of a musical piece. We run this code for an excerpt of the Kraftwerk song “Spacelab,” to demonstrate how this library can provide useful information related to its diatonicity and whole-toneness. As can be seen from the Chromagram in Figure C.4, this song starts in the whole tone scale [F# G# A#C D E], and then moves, at 33 s, to a diatonic set [C D Eb F G Ab Bb C]. In Figure C.5 we show this by plotting the evolution of theTIV.diatonicity, TIV.wholetoneness and TIV.chromaticity outputs for this music.

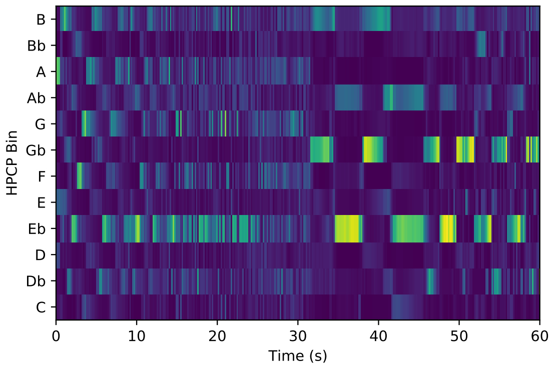


Figure C.4: Chromagram of the first minute of Kraftwerk’s “Spacelab.”

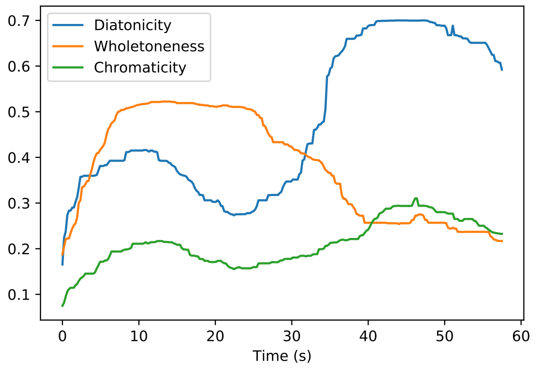


Figure C.5: Output of the diatonicity, chromaticity and whole-toneness harmonic qualities the first minute of Kraftwerk’s “Spacelab”, using the TIV.lib.

Several functions of the TIV.lib result from ongoing research and have been evaluated in previous literature (Bernardes et al., 2017b, 2016a, 2017a, 2016b), where the possibility of the TIV space to geometrically and algebraically capture existing spaces of perceptual and music theoretical value, such as Euler and Krumhansl (Bernardes et al., 2016a), were shown. In particular, we highlight our previous work on key recognition (Bernardes et al., 2017a) and harmonic mixing (Bernardes et al., 2017b; Maçãs et al., 2019), where TIV-based approaches outperformed more traditionally used harmonic features. In addition, the use of TIVs can also extend content-based audio processing by providing a vector space where distances and metrics (e.g., dissonance and harmonic proximity) among multi-level pitch, chords, and keys, capture perceptual aspects of musical phenomena. Examples of creative possibilities of the TIV space have also been shown in Musikverb (Pereira et al., 2018), where it was used for developing a novel type of harmonically-adaptive reverb effect.

We strongly believe that the properties of the TIV space can be further explored in content-based audio processing. For example, the possibility to isolate the harmonic quality in TIV.mag as a pitch-invariant audio representation can be relevant for several MIR tasks that rely on multiple transposed versions of a given musical pattern, such as in query-by-humming, and cover song detection. Moreover, the possibility to compute TIV mixes as a computationally efficient linear combination allows for the fast retrieval of musical audio from large datasets (e.g., Freesound (Font et al., 2013)), as a simple nearest-neighbour search problem. Finally, the newly proposed indicators of tonal quality such as TIV.chromaticity, TIV.diatonicity, TIV.wholte-toneness, and TIV.diss not only extend musical theoretical methodologies to content-based processing from audio performance data, but can also promote a greater understanding of tonal content in MIR tasks.

By providing streamlined access to a set of music theoretic properties which are non-trivial to obtain from commonly used time-frequency representations in MIR such as the STFT (or even from chroma-like representations directly), we believe the TIV.lib can lay the foundation for a kind of "enhanced" MIR in tasks such as chord recognition and key estimation which can directly leverage the complementary contextual information contained within the TIV.lib descriptors.

C.6 Conclusions

In this appendix we have introduced the open-source tool, TIV.lib, as a means to drive the uptake and usage of the Tonal Interval Space both in offline music signal analysis via the python implementation, as well as in online contexts using Pure Data. While we hope to see a growth of applications which benefit from access to music theoretic harmonic features provided by TIV.lib our own future work will focus in two principal areas: i) investigating the processing stages which directly precede the calculation of the TIV; and ii) in the application of the TIV across large datasets. More specifically, we seek to study the impact of different methods for calculating the requisite chroma vectors (e.g., HPCP (Gómez, 2006b), NNLS (Mauch & Dixon, 2010), or timbre-invariant chroma (Muller & Ewert, 2010)) in the TIV space, as pursued in Bernardes et al. (2017a, 2016b) within the scope of audio key detection. Furthermore, we will study an optimal strategy to define the weights, $w_a(k)$, for particular audio sources and to implement the descriptors in a large online musical database supported by content-based analysis, as a strategy to study the descriptors under a large-scale environment for musical retrieval and creation. Finally, we intend to add this library to existing musical audio analysis libraries such as Essentia and Librosa.

Appendix D

Freesound API: add 400k+ sounds to your plugin!

Freesound is a collaborative database where users share sound effects, field recordings, musical samples and other audio material under Creative Commons licenses. Freesound currently contains more than 415k sounds that have been downloaded 139M times by 9M registered users. Freesound offers both a website to interact with the database and a RESTful API which provides programmatic browsing and retrieving of sounds and other Freesound content. We present a JUCE client library which permits an easy integration of Freesound in JUCE projects. The presented library allows, among other things, to make use of the advanced text and audio-based Freesound search engine, to download and upload sounds, and the retrieval of a variety sound analysis information (i.e. audio features) from all Freesound content

Freesound was created in 2005 at the Music Technology Group (MTG) of Universitat Pompeu Fabra (UPF) with the aim of creating a huge collaborative database of sounds released under Creative Commons licenses. The sounds in Freesound are uploaded by its users, who tag and describe them, leading to a huge diversity of sounds. Besides the sound sharing between creators, one of the main aims of Freesound is making this database available for scientific research and for reuse in third party creative applications. The audio content in the repository is analysed using the open-source audio analysis tool Essentia, as well as with the Audio Commons Audio Extractor. More than 100 papers have so far mentioned or used Freesound data for the research, enabling new ways to search the collection and to better characterise the sounds contained. Some of these algorithms are implemented in the Freesound API, an interface to Freesound which allows the integration of Freesound in third party applications. Some of the projects which integrate Freesound content are listed in Freesound Labs[82]. Companies using the Freesound API include, among others,

[82] labs.freesound.org

Waves Audio, Acoustica, Audiogaming, Soundtrap,Soundly, Ardour, Mozilla, Google, and Wolfram.

Freesound hosts a large variety of sounds which can be used for many applications such as music making, sound design, film making or video game development. This sound diversity can be seen from the most used tags in 2018 which include varied terms such as field-recording, ambient, synth, loop, water, beat or voice. A tag-cloud from the uploaded sounds and of the downloaded sounds can be seen in the blog[83]. Furthermore, allowing only Creative Commons licenses makes Freesound being able to grow with the community. The more users and sounds there are, more opportunities for remixing said sounds are created. The amount of sounds present in Freesound, together with the analysis and search tools it provides, enable the creation of novel creative applications. Some selected examples of this are the Freesound Explorer[84], where sounds from Freesound are organised in a 2-dimensional space according to their characteristics; Freemix[85], a web based app for smartphones which uses movement to trigger and modify sounds; Multi Web Audio Sequencer[86], an application for segment-based sequencing of Freesound sound clips; and AudioTexture Free, a plugin that allows to generate infinite sounds, variations or loops based on Freesound content. More of these examples can be found in the aforementioned Freesound Labs portal.

The methods provided by the API can be divided in functions related to searching, sounds, users and packs. These are described in the API documentation [87].The search resources include both a text search which allows exploring sounds by matching their tags and other metadata, and an audio-based search for finding sounds based on their audio characteristics. Search results can be filtered by specifying properties that sounds should match such as its tags, its file format, license, duration, or even high level analysis of the sounds such as its loudness, its hardness, its tempo, its “boominess” or if it is a single event or a loop. An example of how some of these features can be used for searching Freesound is presented in the Audio Commons Audio Extractor Demonstrator[88]. The sounds returned in the search results are accompanied with its metadata, including its analysis, the download link and general information such as its name, license, rating or number of downloads. This comprises the sound resources, together with several methods for interacting with the sounds. This include searching for sounds similar to a specific sound, downloading, rating, commenting, uploading and describing sounds. The user resources allow get-

[83]https://blog.freesound.org/?p=942
[84]https://labs.freesound.org/apps/freesound-explorer.html
[85]https://labs.freesound.org/apps/freemix.html
[86]https://labs.freesound.org/apps/multi-web-audio-sequencer.html
[87]https://freesound.org/docs/api/overview.html
[88]audiocommons.org/ac-audio-extractor/web_demonstrator/

ting information about a user, its uploaded sounds and packs. Finally, the pack resources retrieve the sounds in a pack and allow downloading all the sounds in a single zip file.

The JUCE client library we developed provides access to the Freesound API including all of the aforementioned resources in easy to use C++ classes. The authorization procedure is implemented in the FreesoundClient class and includes methods for carrying out the OAuth2 authentication procedure either by opening an external browser window or by using a component which opens the Freesound login page inside the JUCE application. The API resources which return lists of sounds are automatically parsed to the SoundList class, which contains methods for navigating through the search result pages and to create a FSSound objects for each sound. The FSSound class contains implementations for all the methods contained in the sound resources, from getting the general metadata and downloading the sound to obtaining sound analysis information.

The client library is implemented as a single header and cpp file which can both be easily integrated in JUCE projects as an include file or as a user module. The code is open source and available in a GitHub repository[89]. Together with the header and cpp files, code examples are provided to illustrate how the client can be used. The full documentation is available at the following link.[90]

[89] https://github.com/aframires/freesound-juce
[90] https://mtg.github.io/freesound-juce/index.html

Bibliography

Ableton (2022). Ableton live 10: Compare editions. https://www.ableton.com/en/live/compare-editions/ Acessed in 9/10/2022. [Cited on page 5.]

Akkermans, V., Font Corbera, F., Funollet, J., De Jong, B., Roma Trepat, G., Togias, S., & Serra, X. (2011). Freesound 2: An improved platform for sharing audio clips. In A. Klapuri & Colby Leider (Eds.) *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011,*. University of Miami. [Cited on page 15.]

Alain, G., Chevalier-Boisvert, M., Osterrath, F., & Piche-Taillefer, R. (2020). DeepDrummer: Generating Drum Loops using Deep Learning and a Human in the Loop. In *Proceedings of The 2020 Joint Conference on AI Music Creativity (CSMC + MuMe)*, pp. 81–91. Royal Institute of Technology (KTH). [Cited on page 60.]

American National Standards Institute. Committee on Bioacoustics, Sonn, M., & Acoustical Society of America (1973). *American National Standard Psychoacoustical Terminology*. ANSI S3.20-1973. New York, NY: American National Standards Institute. [Cited on page 23.]

Amiot, E. (2016). *Music Through Fourier Space: Discrete Fourier Transform in Music Theory*. Cham, Switzerland: Springer. [Cited on pages 145, 146, and 148.]

Andersen, K. & Knees, P. (2016). Conversations with Expert Users in Music Retrieval and Research Challenges for Creative MIR. In J. Devaney, M. I. Mandel, D. Turnbull, & G. Tzanetakis (Eds.) *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pp. 122–128. ISMIR. [Cited on page 9.]

Anderson, E. J. (1997). *Limitations of short-time Fourier transforms in polyphonic pitch recognition*. PhD Thesis, University of Washington, Seattle, WA. [Cited on page 22.]

Aouameur, C., Esling, P., & Hadjeres, G. (2019). Neural Drum Machine : An Interactive System for Real-time Synthesis of Drum Sounds. In K. Grace, M. Cook, D. Ventura, & M. L. Maher (Eds.) *Proceedings of the Tenth International Conference on Computational Creativity, ICCC 2019*, pp. 92–99. Association for Computational Creativity (ACC). [Cited on pages 92 and 96.]

Arjovsky, M. & Bottou, L. (2017). Towards Principled Methods for Training Generative Adversarial Networks. In *5th International Conference on Learning Representations, ICLR 2017 Conference Track Proceedings*, pp. 1–17. OpenReview.net. [Cited on page 88.]

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In D. Precup & Y. Whye Teh (Eds.) *Proceedings of the 34th International Conference on Machine Learning, ICML'17, Proceedings of Machine Learning Research (PMLR)*, vol. PMLR 70, pp. 214–223. JMLR.org. [Cited on page 88.]

Bello, J. P., Ravelli, E., & Sandler, M. B. (2006). Drum Sound Analysis for the Manipulation of Rhythm in Drum Loops. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, pp. V–V. IEEE. [Cited on page 46.]

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2018). Automatic Music Transcription: An Overview. *IEEE Signal Processing Magazine*, *36*(1), 20–30. [Cited on page 144.]

Benetos, E., Kotti, M., & Kotropoulos, C. (2006). Musical Instrument Classification using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, pp. V–V. IEEE. [Cited on page 33.]

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127. [Cited on pages 28 and 29.]

Bernardes, G., Cocharro, D., Caetano, M., Guedes, C., & Davies, M. E. (2016a). A multi-level tonal interval space for modelling pitch relatedness and musical consonance. *Journal of New Music Research*, *45*(4), 281–294. [Cited on pages 14, 144, 145, 146, 148, 150, 152, and 153.]

Bernardes, G., Cocharro, D., Guedes, C., & Davies, M. E. P. (2016b). Harmony Generation Driven by a Perceptually Motivated Tonal Interval Space. *Computers in Entertainment*, *14*(2), 6:1–6:21. [Cited on pages 152 and 154.]

Bernardes, G., Davies, M. E. P., & Guedes, C. (2017a). Automatic musical key estimation with adaptive mode bias. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 316–320. IEEE. [Cited on pages 151, 152, 153, and 154.]

Bernardes, G., Davies, M. E. P., & Guedes, C. (2017b). A Hierarchical Harmonic Mixing Method. In M. Aramaki, M. E. P. Davies, R. Kronland-Martinet, & S. Ystad (Eds.) *Music Technology with Swing, CMMR 2017*, *Lecture Notes in Computer Science (LNCS)*, vol. 11265, pp. 151–170. Cham: Springer International Publishing. [Cited on pages 144, 146, 150, 152, and 153.]

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The Million Song Dataset. In A. Klapuri & C. Leider (Eds.) *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, pp. 591–596. University of Miami. [Cited on page 20.]

Bitton, A., Esling, P., Caillon, A., & Fouilleul, M. (2019). Assisted Sound Sample Generation with Musical Conditioning in Adversarial Auto-Encoders. Http://arxiv.org/abs/1904.06215. [Cited on page 92.]

Bitton, A., Esling, P., & Chemla-Romeu-Santos, A. (2018). Modulated Variational auto-Encoders for many-to-many musical timbre transfer. Http://arxiv.org/abs/1810.00222. [Cited on page 92.]

Bińkowski, M., Sutherland, D. J., Arbel, M., & Gretton, A. (2018). Demystifying MMD GANs. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*, pp. 1–36. ICLR. [Cited on page 111.]

Blaauw, M. & Bonada, J. (2017). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. *Applied Sciences*, *7*(12), 1313. [Cited on page 106.]

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. R., & Serra, X. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. In A. de Souza Britto Junior, F. Gouyon, & S. Dixon (Eds.) *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pp. 493–498. ISMIR. [Cited on pages 27, 51, 56, 97, 143, and 144.]

Bosch, J. J., Janer, J., Fuhrmann, F., & Herrera, P. (2012). A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In F. Gouyon, P. Herrera, L. G. Martins, & M. Müller (Eds.) *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pp. 559–564. ISMIR. [Cited on page 34.]

Brunner, G., Konrad, A., Wang, Y., & Wattenhofer, R. (2018). MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. In E. Gómez, X. Hu, E. Humphrey, & E. Benetos (Eds.) *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 747–754. ISMIR. [Cited on page 80.]

Butler, M. J. (2006). *Unlocking the Groove: Rhythm, meter, and musical design in electronic dance music.* Bloomington, IN: Indiana University Press. [Cited on page 1.]

Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., & Widmer, G. (2016). madmom: A New Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM international conference on Multimedia, MM '16*, pp. 1174–1178. Association for Computing Machinery. [Cited on page 143.]

Böck, S., Krebs, F., & Widmer, G. (2015). Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters. In M. Müller & F. Wiering (Eds.) *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pp. 657–663. ISMIR. [Cited on pages 21 and 55.]

Carr, C. J. & Zukowski, Z. (2018). Generating Albums with SampleRNN to Imitate Metal, Rock, and Punk Bands. Http://arxiv.org/abs/1811.06633. [Cited on pages 91, 105, and 106.]

Chandna, P., Blaauw, M., Bonada, J., & Gómez, E. (2019). WGANSing: A Multi-Voice Singing Voice Synthesizer Based on the Wasserstein-GAN. In *27th European Signal Processing Conference, EUSIPCO 2019*, pp. 1–5. IEEE. [Cited on pages 98 and 109.]

Chandna, P., Ramires, A., Serra, X., & Gómez, E. (2021). Loopnet: Musical Loop Synthesis Conditioned on Intuitive Musical Parameters. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021*, pp. 3395–3399. IEEE. [Cited on pages 14 and 60.]

Chen, B.-Y., Smith, J. B., & Yang, Y.-H. (2020). Neural loop combiner: Neural network models for assessing the compatibility of loops. In Julie Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, & T. de Reuse (Eds.) *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*, pp. 424–431. ISMIR. [Cited on page 60.]

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16*, pp. 2180–2188. Curran Associates Inc. [Cited on page 90.]

Cherry, E. & Latulipe, C. (2014). Quantifying the Creativity Support of Digital Tools through the Creativity Support Index. *ACM Transactions on Computer-Human Interaction*, *21*(4), 21:1–21:25. [Cited on page 116.]

Chew, E. (2000). *Towards a mathematical model of tonality.* PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA. [Cited on page 22.]

Chew, E. (2007). Out of the Grid and Into the Spiral: Geometric Interpretations of and Comparisons with the Spiral-Array Model. In W. B. Hewlett, E. Selfridge-Field, & E. J. Correia (Eds.) *Tonal Theory for the Digital Age (Computing in Musicology)*, vol. 15, pp. 51–72. Stanford, CA: Center for Computer Assisted Research in the Humanities, Stanford University (CCARH). [Cited on page 145.]

Ching, J., Ramires, A., & Yang, Y.-H. (2020). Instrument Role Classification: Auto-tagging for Loop Based Music. In *Proceedings of The 2020 Joint Conference on AI Music Creativity (CSMC + MuMe)*, pp. 196–202. Royal Institute of Technology (KTH). [Cited on pages 63 and 65.]

Chuan, C.-H. & Chew, E. (2005). Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm. In *2005 IEEE International Conference on Multimedia and Expo, ICME 2005*, pp. 21–24. IEEE. [Cited on page 145.]

Cífka, O. (2021). *Deep learning methods for music style transfer. (Méthodes d'apprentissage profond pour le transfert de style musical).* PhD Thesis, Polytechnic Institute of Paris, Palaiseau, France. [Cited on page 80.]

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In Y. Bengio & Y. LeCun (Eds.) *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings.* [Cited on pages 40 and 63.]

Cocharro, D., Sioros, G., Caetano, M. F., & Davies, M. E. P. (2014). Real-time Manipulation of Syncopation in Audio Loops. In A. Georgaki & G. Kouroupetroglou (Eds.) *Joint Proceedings of the 40th International Computer Music Conference, ICMC 2014, and the 11th Sound and Music Computing Conference, SMC 2014.* Michigan Publishing. [Cited on page 60.]

Creswell, A. & Bharath, A. A. (2019). Inverting the Generator of a Generative Adversarial Network. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(7), 1967–1974. [Cited on page 89.]

Cuesta, H., McFee, B., & Gómez, E. (2020). Multiple F0 Estimation in Vocal Ensembles using Convolutional Neural Networks. In J. Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, & T. de Reuse (Eds.) *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*, pp. 302–309. ISMIR. [Cited on page 22.]

Curtis Hawthorne, Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., & Eck, D. (2018). Onsets and frames: Dual-objective piano transcription. In E. Gómez, X. Hu, E. Humphrey, & E. Benetos (Eds.) *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 50–57. ISMIR. [Cited on page 144.]

Davies, M. E., Hamel, P., Yoshii, K., Goto, M., de Souza Britto Junior, A., Fabien Gouyon, & Dixon, S. (2013). AutoMashUpper: An automatic multi-song mashup system. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pp. 575–580. ISMIR. [Cited on pages 61 and 74.]

Davies, M. E. & Plumbley, M. D. (2008). Exploring the effect of rhythmic style classification on automatic tempo estimation. In *16th European Signal Processing Conference, EUSIPCO 2008*, pp. 1–5. IEEE. [Cited on page 21.]

Davies, M. E. P. & Plumbley, M. D. (2007). Context-Dependent Beat Tracking of Musical Audio. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(3), 1009–1020. [Cited on page 21.]

Davis, J. & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, pp. 233–240. Association for Computing Machinery. [Cited on page 70.]

Dawson, M. R. W., Perez, A., & Sylvestre, S. (2020). Artificial Neural Networks Solve Musical Problems With Fourier Phase Spaces. *Scientific Reports*, *10*(1), 7151. [Cited on page 145.]

De Haas, B. W., Veltkamp, R. C., & Wiering, F. (2008). Tonal Pitch Step Distance: a Similarity Measure for Chord Progressions. In J. P. Bello, E. Chew, & D. Turnbull (Eds.) *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008*, pp. 51–56. ISMIR. [Cited on page 145.]

Degara, N., Rua, E. A., Pena, A., Torres-Guijarro, S., Davies, M. E. P., & Plumbley, M. D. (2012). Reliability-Informed Beat Tracking of Musical Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 290–301. [Cited on page 55.]

Dicale, B. (2007). Jean-michel jarre re-records oxygène. http://www1.rfi.fr/musiqueen/articles/096/article_7986.asp/ Acessed in 9/10/2022. [Cited on page 3.]

Dieleman, S., van den Oord, A., & Simonyan, K. (2018). The Challenge of Realistic Music Generation: Modelling Raw Audio at Scale. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18*, pp. 8000–8010. Red Hook, NY: Curran Associates Inc. [Cited on pages 106 and 111.]

Dixon, S. (2001). Automatic Extraction of Tempo and Beat From Expressive Performances. *Journal of New Music Research*, *30*(1), 39–58. [Cited on page 21.]

Donahue, C., McAuley, J. J., & Puckette, M. S. (2019). Adversarial Audio Synthesis. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, pp. 1–16. ICLR. [Cited on pages 91, 92, and 106.]

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., & Yang, Y.-H. (2018). MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment. In S. A. McIlraith & K. Q. Weinberger (Eds.) *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pp. 34–41. Palo Alto, CA: AAAI Press. [Cited on page 80.]

Downie, S. J. (2003). Music information retrieval. *Annual Review of Information Science and Technology*, *37*(1), 295–340. [Cited on page 11.]

Drysdale, J., Ramires, A., Serra, X., & Hockman, J. (2022). Improved Automatic Instrumentation Role Classification and Loop Activation Transcription. In G. Evangelista & N. Holighaus (Eds.) *Proceedings of the 25th International Conference on Digital Audio Effects, DAFx2022*, *DAFx20' sViennaSeries*, vol. 3, pp. 264–271. DAFx. [Cited on page 13.]

Drysdale, J., Tomczak, M., & Hockman, J. (2020). Adversarial synthesis of drum sounds. In G. Evangelista (Ed.) *Proceedings of the 23rd International Conference on Digital Audio Effects, eDAFx-2020*, *DAFx20' sViennaSeries*, vol. 1, pp. 167–172. DAFx. [Cited on pages 92 and 116.]

Drysdale, J., Tomczak, M., & Hockman, J. (2021). Style-based Drum Synthesis with GAN Inversion. In *Extended Abstracts for the Late-Breaking Demo Sessions of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021*. ISMIR. [Cited on page 116.]

Défossez, A. (2021). Hybrid Spectrogram and Waveform Source Separation. In *Proceedings of the ISMIR 2021 Workshop on Music Source Separation: Music Demixing Workshop, MDX21*, pp. 1–11. ISMIR. [Cited on page 80.]

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). GANSynth: Adversarial Neural Audio Synthesis. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*. ICLR. [Cited on pages 91 and 92.]

Engel, J., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., & Simonyan, K. (2017). Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In D. Precup & Y. W. Teh (Eds.) *Proceedings of the*

34th International Conference on Machine Learning, ICML'17, Proceedings of Machine Learning Research (PMLR), vol. PMLR 70, pp. 1068–1077. JMLR.org. [Cited on pages 34, 36, 37, 40, 80, 91, 92, 106, and 116.]

Engel, J. H., Hantrakul, L., Gu, C., & Roberts, A. (2020). DDSP: Differentiable Digital Signal Processing. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*. ICLR. [Cited on pages 80, 110, and 113.]

Esling, P. & Bitton, A. (2018). Bridging Audio Analysis, Perception and Synthesis with Perceptually-regularized Variational Timbre Spaces. In E. Gómez, X. Hu, E. Humphrey, & E. Benetos (Eds.) *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 175–181. ISMIR. [Cited on pages 90, 91, and 92.]

Essid, S., Richard, G., & David, B. (2006). Musical instrument recognition by pairwise classification strategies. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(4), 1401–1412. [Cited on page 33.]

Faraldo, ., Gómez, E., Jordà, S., & Herrera, P. (2016). Key Estimation in Electronic Dance Music. In N. Ferro, F. Crestan, M.-F. Moens, J. Mothe, F. Silvestri, G. M. Nunzio, C. Hauff, & G. Silvello (Eds.) *Proceedings of the 38th European Conference on Information Retrieval, ECIR 2016: Advances in Information Retrieva, Lecture Notes in Computer Science (LNCS)*, vol. 9626, pp. 335–347. Springer. [Cited on pages 22, 51, and 56.]

Faraldo Pérez, Á. (2018). *Tonality Estimation in Electronic Dance Music: A Computational and Musically Informed Examination.* PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 22.]

Favory, X., Font, F., & Serra, X. (2020). Search Result Clustering in Collaborative Sound Collections. In C. Gurrin, B. . Jónsson, N. Kando, K. Schöffmann, Y.-P. P. Chen, & N. E. O'Connor (Eds.) *Proceedings of the 2020 on International Conference on Multimedia Retrieval, ICMR 2020*, pp. 207–214. ACM. [Cited on page 11.]

Feinstein, A. R. & Cicchetti, D. V. (1990). High agreement but low Kappa: I. the problems of two paradoxes. *Journal of Clinical Epidemiology*, *43*(6), 543–549. [Cited on page 53.]

FitzGerald, D., Cranitch, M., & Coyle, E. (2006). Sound Source Separation Using Shifted Non-Negative Tensor Factorisation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, ICASSP 2006*, vol. 5, pp. V–V. IEEE. [Cited on page 60.]

Fonseca, E., Plakal, M., Ellis, D. P. W., Font, F., Favory, X., & Serra, X. (2019). Learning Sound Event Classifiers from Web Audio with Noisy Labels. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019*, pp. 21–25. IEEE. [Cited on page 20.]

Font, F. & Bandiera, G. (2017). Freesound Explorer: Make Music While Discovering Freesound! In F. Thalmann & S. Ewert (Eds.) *Proceedings of the 3rd International Web Audio Conference, WAC 2017*. Queen Mary University of London. [Cited on page 11.]

Font, F., Roma, G., & Serra, X. (2013). Freesound Technical Demo. In *Proceedings of the 21st ACM international conference on Multimedia, MM '13*, pp. 411–412. ACM. [Cited on pages 6, 47, 66, 96, 97, 123, and 153.]

Font, F. & Serra, X. (2016). Tempo Estimation for Music Loops and a Simple Confidence Measure. In J. Devaney, M. I. Mandel, D. Turnbull, & G. Tzanetakis (Eds.) *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pp. 269–275. ISMIR. [Cited on pages 47, 48, 51, 55, and 107.]

Forte, A. (1964). A Theory of Set-Complexes for Music. *Journal of Music Theory*, *8*(2), 136–183. [Cited on page 148.]

Forte, A. (1973). *The structure of atonal music*, vol. 304. New Haven, CT: Yale University Press. [Cited on page 148.]

Fuhrmann, F. (2012). *Automatic musical instrument recognition from polyphonic music audio signals*. PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 33.]

Fuhrmann, F. & Herrera, P. (2010). Polyphonic instrument recognition for exploring semantic similarities in music. In *Proceedings of the 13th International Conference on Digital Audio Effects, DAFx-10*, pp. 1–8. [Cited on page 33.]

Fujishima, T. (1999). Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music. In *Proceedings of the 25th International Computer Music Conference, ICMC 1999*, pp. 464–467. Michigan Publishing. [Cited on page 144.]

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, C. R., Plakal, M., & Ritter, M. (2017). Audio Set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017*, pp. 776–780. IEEE. [Cited on page 111.]

Gillet, O. & Richard, G. (2004). Automatic transcription of drum loops. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. iv–iv. IEEE. [Cited on page 46.]

Gillet, O. & Richard, G. (2005). Drum Loops Retrieval from Spoken Queries. *Journal of Intelligent Information Systems*, *24*(2), 159–177. [Cited on pages 46 and 60.]

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh & M. Titterington (Eds.) *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, AISTATS 2009*, *Proceedings of Machine Learning Research (PMLR)*, vol. PMLR 9, pp. 249–256. JMLR.org. [Cited on page 31.]

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv. [Cited on pages 83 and 84.]

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridg, MA: MIT Press. [Cited on pages 27, 28, 31, and 32.]

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.) *Advances in neural information processing systems 27: Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, pp. 2672–2680. Red Hook, NY: Curran Associates, Inc. [Cited on pages 80, 87, and 116.]

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2002). RWC Music Database: Popular, Classical, and Jazz Music Databases. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference, ISMIR 2002*, pp. 287–288. ISMIR. [Cited on page 34.]

Gouyon, F. & Dixon, S. (2005). A Review of Automatic Rhythm Description Systems. *Computer Music Journal*, *29*(1), 34–54. [Cited on page 21.]

Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., Uhle, C., & Cano, P. (2006). An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(5), 1832–1844. [Cited on pages 21, 46, and 55.]

Grachten, M., Lattner, S., & Deruty, E. (2020). BassNet: A Variational Gated Autoencoder for Conditional Generation of Bass Guitar Tracks with Learned Interactive Control. *Applied Sciences*, *10*(18). [Cited on page 80.]

Griffin, D. & Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *32*(2), 236–243. [Cited on pages 111 and 112.]

Grover, A., Dhar, M., & Ermon, S. (2018). Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models. In S. A. McIlraith & K. Q. Weinberger (Eds.) *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pp. 3069–3076. AAAI Press. [Cited on page 88.]

Gómez, E. (2006a). *Tonal Description of Music Audio Signals.* PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 22.]

Gómez, E. (2006b). Tonal Description of Polyphonic Audio for Music Content Processing. *INFORMS Journal on Computing*, *18*(3), 294–304. [Cited on pages 56, 109, 144, and 154.]

Gómez-Marín, D., Jordà, S., & Herrera, P. (2015). PAD and SAD: Two awareness-weighted rhythmic similarity distances. In Meinard Müller & F. Wiering (Eds.) *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pp. 666–672. ISMIR. [Cited on page 47.]

Han, Y., Kim, J., & Lee, K. (2017). Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(1), 208–221. [Cited on pages 33, 34, 36, and 40.]

Harte, C., Sandler, M., & Gasser, M. (2006). Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia, AMCMM '06*, pp. 21–26. ACM. [Cited on page 150.]

Hasnain, Z. (2017). How the roland tr-808 revolutionized music. https://www.theverge.com/2017/4/3/15162488/roland-tr-808-music-drum-machine-revolutionized-music/ Acessed in 9/10/2022. [Cited on page 78.]

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*, pp. 1026–1034. IEEE. [Cited on pages 31 and 64.]

Heise, S., Hlatky, M., & Loviscach, J. (2008). SoundTorch: Quick Browsing in Large Audio Collections. In *Proceedings of the125th Audio Engineering Society Convention 2008, AES 2008*, p. Paper 7544. AES. [Cited on page 11.]

Herrera-Boyer, P., Peeters, G., & Dubnov, S. (2003). Automatic Classification of Musical Instrument Sounds. *Journal of New Music Research*, *32*(1), 3–21. [Cited on pages 23 and 33.]

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507. [Cited on page 86.]

Hoffman, J. (2008). On Pitch-Class Set Cartography: Relations between Voice-Leading Spaces and Fourier Spaces. *Journal of Music Theory*, *52*(2), 219–249. [Cited on page 148.]

Holmes, T. (2012). *Electronic and experimental music: technology, music, and culture*. New York, NY: Routledge, 4th edn. [Cited on pages 2 and 4.]

Huron, D. (1994). Interval-class content in equally tempered pitch-class sets: Common scales exhibit optimum tonal consonance. *Music Perception*, *11*(3), 289–305. [Cited on page 146.]

Härkönen, E., Hertzmann, A., Lehtinen, J., & Paris, S. (2020). GANSpace: Discovering Interpretable GAN Controls. In H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, & H.-T. Lin (Eds.) *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, pp. 1–10. Curran Associates, Inc. [Cited on pages 89 and 119.]

Ioffe, S. & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning, July 6 - 11, 2015, ICML'15*, vol. 37, pp. 448–456. JMLR.org. [Cited on pages 40 and 63.]

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 5967–5976. IEEE. [Cited on pages 89 and 90.]

Jahanian, A., Chai, L., & Isola, P. (2020). On the "steerability" of generative adversarial networks. In *8th International Conference on Learning Representations, ICLR 2020, Conference Track Proceedings*. ICLR. [Cited on page 89.]

Jordan, M. I. & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260. [Cited on pages 25 and 26.]

Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, *61*, 101236. [Cited on page 19.]

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International*

Conference on Learning Representations, ICLR 2018, Conference Track Proceedings. ICLR. [Cited on pages 88, 93, and 117.]

Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2019*, pp. 4401–4410. CVF / IEEE. [Cited on pages 117 and 119.]

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and Improving the Image Quality of StyleGAN. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pp. 8107–8116. CVF / IEEE. [Cited on pages 88, 116, and 118.]

Kilgour, K., Zuluaga, M., Roblek, D., & Sharifi, M. (2019). Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association, INTERSPEECH 2019*, pp. 2350–2354. ISCA. [Cited on pages 111 and 122.]

Kingma, D. P. & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. ICLR. [Cited on pages 41, 64, 100, and 111.]

Kingma, D. P. & Welling, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*. ICLR. [Cited on pages 26, 80, 86, and 116.]

Knees, P. & Schedl, M. (2016). *Music Similarity and Retrieval: An Introduction to Audio- and Web-Based Strategies*, *The Information Retrieval Series*, vol. 36. Heidelberg: Springer-Verlag GmbH, 1st edn. [Cited on pages 20 and 23.]

Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., & Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017*, pp. 5220–5224. IEEE. [Cited on page 37.]

Krumhans, C. L. (1990). *Cognitive Foundations of Musical Pitchl*. Oxford Psychology Series No. 17. New York, NY: Oxford University Press. [Cited on page 56.]

Laguarta, J., Hueto, F., & Subirana, B. (2020). COVID-19 Artificial Intelligence Diagnosis Using Only Cough Recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, *1*, 275–281. [Cited on page 19.]

Latham, A. (Ed.) (2011). *The Oxford Companion to Music*. Oxford, UK: Oxford University Press. [Cited on page 78.]

Lattner, S. & Grachten, M. (2019). High-Level Control of Drum Track Generation Using Learned Patterns of Rhythmic Interaction. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA 2019*, pp. 35–39. IEEE. [Cited on page 80.]

Law, E., West, K., Mandel, M. I., Bay, M., & Downie, J. S. (2009). Evaluation of algorithms using games: The case of music tagging. In K. Hirata, G. Tzanetakis, & K. Yoshii (Eds.) *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pp. 387–392. ISMIR. [Cited on page 20.]

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. [Cited on page 36.]

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. [Cited on page 90.]

Lerdahl, F. (2004). *Tonal pitch space.* Oxford, UK: Oxford University Press. [Cited on page 145.]

Lessig, L. (2008). *Remix: Making art and commerce thrive in the hybrid economy.* New York, NY: Penguin Press. [Cited on page 5.]

Li, P., Qian, J., & Wang, T. (2015). Automatic Instrument Recognition in Polyphonic Music Using Convolutional Neural Networks. Http://arxiv.org/abs/1511.05520. [Cited on pages 33 and 36.]

Liu, L. (2005). *The Chinese Neolithic: Trajectories to Early States.* New Studies in Archaeology. Cambridge, UK: Cambridge University Press. [Cited on page 77.]

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep Learning Face Attributes in the Wild. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*, pp. 3730–3738. IEEE. [Cited on page 90.]

López-Serrano, P., Dittmar, C., Driedger, J., & Müller, M. (2016). Towards Modeling and Decomposing Loop-Based Electronic Music. In J. Devaney, M. I. Mandel, D. Turnbull, & G. Tzanetakis (Eds.) *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pp. 502–508. ISMIR. [Cited on pages 60, 67, 69, and 71.]

López-Serrano, P., Dittmar, C., & Müller, M. (2017). Finding Drum Breaks in Digital Music Recordings. In M. Aramaki, M. E. P. Davies, R. Kronland-Martinet, & S. Ystad (Eds.) *Music Technology with Swing, CMMR 2017*, *Lecture Notes in Computer Science (LNCS)*, vol. 11265, pp. 111–122. Springer International Publishing. [Cited on pages 60 and 73.]

Manning, P. (2004). *Electronic and Computer Music.* Oxford, UK: Oxford University Press. [Cited on page 3.]

Mauch, M. & Dixon, S. (2010). Approximate Note Transcription for the Improved Identification of Difficult Chords. In S. J. Downie & R. C. Veltkamp (Eds.) *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, pp. 135–140. ISMIR. [Cited on pages 144 and 154.]

Mauch, M. & Dixon, S. (2014). pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014*, pp. 659–663. IEEE. [Cited on page 22.]

Maçãs, C., Rodrigues, A., Bernardes, G., & Machado, P. (2019). MixMash: An Assistive Tool for Music Mashup Creation from Large Music Collections. *International Journal of Art, Culture, Design, and Technology (IJACDT)*, *8*(2), 20–40. [Cited on page 153.]

McFee, B., Humphrey, E. J., & Bello, J. P. (2015a). A software framework for musical data augmentation. In M. Müller & F. Wiering (Eds.) *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pp. 248–254. ISMIR. [Cited on pages 36 and 37.]

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015b). Librosa: Audio and music signal analysis in python. In Kathryn Huff & J. Bergstra (Eds.) *Proceedings of the 14th Python in Science Conference, SciPy 2015*, pp. 8–24. SciPy Conferences. [Cited on page 143.]

McFee, B., Salamon, J., & Bello, J. P. (2018). Adaptive Pooling Operators for Weakly Labeled Sound Event Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(11), 2180–2193. [Cited on page 64.]

McKinney, M. F., Moelants, D., Davies, M. E., & Klapuri, A. (2007). Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. *Journal of New Music Research*, *36*(1), 1–16. [Cited on page 21.]

McVicar, M., Santos-Rodríguez, R., Ni, Y., & Bie, T. D. (2014). Automatic Chord Estimation from Audio: A Review of the State of the Art. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(2), 556–575. [Cited on page 23.]

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys*, *54*(6), 115:1–115:35. [Cited on page 33.]

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. C., & Bengio, Y. (2017). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017*. ICLR. [Cited on pages 80, 106, and 116.]

Miron, M., Davies, M. E., & Gouyon, F. (2013). An open-source drum transcription system for Pure Data and Max MSP. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 221–225. IEEE. [Cited on page 109.]

Mirza, M. & Osindero, S. (2014). Conditional Generative Adversarial Nets. Http://arxiv.org/abs/1411.1784. [Cited on page 90.]

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. New York, NY: McGraw-Hill. [Cited on page 27.]

Muller, M. & Ewert, S. (2010). Towards Timbre-Invariant Audio Features for Harmony-Based Music. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 649–662. [Cited on pages 144 and 154.]

Nair, V. & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In J. Fürnkranz & T. Joachims (Eds.) *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, ICML'10, pp. 807–814. Madison, WI: Omnipress. [Cited on page 29.]

Nam, J., Choi, K., Lee, J., Chou, S.-Y., & Yang, Y.-H. (2019). Deep Learning for Audio-Based Music Classification and Tagging: Teaching Computers to Distinguish Rock from Bach. *IEEE Signal Processing Magazine*, *36*(1), 41–51. [Cited on pages 33 and 36.]

Nistal, J., Lattner, S., & Richard, G. (2021). DrumGAN: Synthesis of drum sounds with timbral feature conditioning using Generative Adversarial Networks. In J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, & A. Srinivasamurthy (Eds.) *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2021*, pp. 484–492. ISMIR. [Cited on pages 92, 93, 106, 112, 116, and 123.]

Noland, K. & Sandler, M. (2007). Signal Processing Parameters for Tonality Estimation. In *Proceedings of the 122nd Audio Engineering Society Convention, AES 2007*, p. Paper 7155. AES. [Cited on page 56.]

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., & Kavukcuoglu, K. (2016a). WaveNet: A Generative Model for Raw Audio. In *The 9th ISCA Speech Synthesis Workshop, SSW9*, p. 125. International Speech Communication Association. [Cited on pages 80, 85, 96, 98, 109, and 116.]

Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., & Kavukcuoglu, K. (2016b). Conditional Image Generation with PixelCNN Decoders. In D. D. Lee, U. von Luxburg, R. Garnett, M. Sugiyama, & I. Guyon (Eds.) *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16*, NIPS'16, pp. 4797–4805. Red Hook, NY, USA: Curran Associates Inc. [Cited on page 85.]

Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., & Hassabis, D. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proceedings of the 35th International Conference on Machine Learning, ICML 18*, *Proceedings of Machine Learning Research (PMLR)*, vol. PMLR 80, pp. 3915–3923. JMLR.org. [Cited on page 106.]

Palombini, C. (1993). Machine Songs V: Pierre Schaeffer: From Research into Noises to Experimental Music. *Computer Music Journal*, *17*(3), 14–19. [Cited on page 3.]

Park, T. & Lee, T. (2015). Musical instrument sound classification with deep convolutional neural network using feature fusion approach. Http://arxiv.org/abs/1512.07370. [Cited on pages 33 and 36.]

Pearce, A., Brookes, T., & Mason, R. (2017). Timbral Attributes for Sound Effect Library Searching. In *Proceedings of the International Conference on Semantic Audio, AES 2017*. AES. [Cited on pages 97, 109, and 123.]

Peeters, G. (2006). Musical key estimation of audio signal based on hidden Markov modeling of chroma vectors. In *Proceedings of the 9th International Conference on Digital Audio Effects, DAFx-2006*, p. Paper 127. DAFx. [Cited on page 22.]

Percival, G. & Tzanetakis, G. (2014). Streamlined Tempo Estimation Based on Autocorrelation and Cross-correlation With Pulses. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(12), 1765–1776. [Cited on page 55.]

Pereira, J. P., Bernardes, G., & Penha, R. (2018). Musikverb: A Harmonically Adaptive Audio Reverberation. In M. Davies, A. Ferreira, G. Campos, & N. Fonseca (Eds.) *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx-18)*, pp. 357–360. DAFx. [Cited on page 153.]

Pons, J. (2019). *Deep neural networks for music and audio tagging*. PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 70.]

Pons, J., Nieto, O., Prockup, M., Schmidt, E., Ehmann, A., & Serra, X. (2017a). End-to-end learning for music audio tagging at scale. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pp. 472–477. ISMIR. [Cited on page 65.]

Pons, J., Slizovskaia, O., Gong, R., Gómez, E., & Serra, X. (2017b). Timbre analysis of music audio signals with convolutional neural networks. In *25th European Signal Processing Conference, EUSIPCO 2017*, pp. 2744–2748. EURASIP/ IEEE. [Cited on pages xvii, 33, 34, 36, 37, 40, 41, 61, and 62.]

Quinn, I. (2007). General Equal-Tempered Harmony: Parts 2 and 3. *Perspectives of New Music*, *45*(1), 4–63. [Cited on pages 145 and 148.]

Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., & Ellis, D. P. (2014). mir_eval: A transparent implementation of common MIR metrics. In H.-M. Wang, Y.-H. Yang, & J. H. Lee (Eds.) *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pp. 367–372. ISMIR. [Cited on page 57.]

Ramires, A., Bernardes, G., Davies, M., & Serra, X. (2020a). TIV.LIB: An Open-Source Library for the Tonal Description of Musical Audio. In G. Evangelista (Ed.) *Proceedings of the 23rd International Conference on Digital Audio Effects, eDAFx-2020, DAFx20' sViennaSeries*, vol. 1, pp. 304–309. DAFx. [Cited on page 14.]

Ramires, A., Chandna, P., Favory, X., Gómez, E., & Serra, X. (2020b). Neural Percussive Synthesis Parameterised by High-Level Timbral Features. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020*, pp. 786–790. IEEE. [Cited on pages 14, 92, 106, 112, 123, and 142.]

Ramires, A. & Font, F. (2019). Poster: Freesound API: add 400k+ sounds to your plugin. In *Audio Developer Conference 2019 (ADC19)*. ADC. [Cited on page 14.]

Ramires, A., Font, F., Bogdanov, D., Smith, J. B. L., Yang, Y.-H., Ching, J., Chen, B.-Y., Wu, Y.-K., Wei-Han, H., & Serra, X. (2020c). The Freesound Loop Dataset and Annotation Tool. In J. Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, & T. de Reuse (Eds.) *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2020*, pp. 287–295. ISMIR. [Cited on pages 13 and 66.]

Ramires, A., Juras, J., Parker, J. D., & Serra, X. (2022). A Study of Control Methods for Percussive Sound Synthesis Based on GANs. In G. Evangelista & N. Holighaus (Eds.) *Proceedings of the 25th International Conference on Digital Audio Effects, DAFx2022, DAFx20' sViennaSeries*, vol. 3, pp. 224–231. DAFx. [Cited on page 14.]

Ramires, A. & Serra, X. (2019). Data augmentation for instrument classification robust to audio effects. In *Proceedings of the 22nd International Conference on Digital Audio Effects, DAFx2019*, pp. 1–6. DAFx. [Cited on page 13.]

Ratcliffe, R. (2014). A proposed typology of sampled material within electronic dance music. *Dancecult: Journal of Electronic Dance Music Culture*, *6*(1), 97–122. [Cited on pages 1 and 6.]

Ravelli, E., Bello, J. P., & Sandler, M. (2007). Automatic Rhythm Modification of Drum Loops. *IEEE Signal Processing Letters*, *14*(4), 228–231. [Cited on page 46.]

Razavi, A., Oord, A. v. d., & Vinyals, O. (2019). Generating Diverse High-Fidelity Images with VQ-VAE-2. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 14837–14847. Curran Associates, Inc. [Cited on page 87.]

Reiss, J. D. & McPherson, A. P. (2014). *Audio effects: Theory, implementation and application.* Boca Raton, FL: CRC Press. [Cited on page 39.]

Ritzer, G. & Jurgenson, N. (2010). Production, Consumption, Prosumption: The nature of capitalism in the age of the digital 'prosumer'. *Journal of Consumer Culture*, *10*(1), 13–36. [Cited on page 5.]

Roma, G. (2015). *Algorithms and representations for supporting online music creation with large-scale audio databases.* PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 47.]

Roma, G. & Serra, X. (2015). Querying Freesound with a microphone. In S. Goldszmidt, N. Schnell, V. Saiz, & B. Matuszewski (Eds.) *Proceedings of the 1st International Web Audio Conference, WAC 2015.* IRCAM. [Cited on page 11.]

Romani Picas, O., Parra Rodriguez, H., Dabiri, D., Tokuda, H., Hariya, W., Oishi, K., & Serra, X. (2015). A real-time system for measuring sound goodness in instrumental sounds. In *Proceedings of the 138nd Audio Engineering Society Convention, AES 2015*, p. 9350. AES. [Cited on page 34.]

Sahai, A., Weber, R., & McWilliams, B. (2019). Spectrogram Feature Losses for Music Source Separation. In *27th European Signal Processing Conference (EUSIPCO 2019), 02-06 September 2019, A Coruna, Spain*, pp. 1–5. IEEE. [Cited on page 99.]

Salamon, J. & Bello, J. P. (2017). Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters*, *24*(3), 279–283. [Cited on pages 37 and 42.]

Salamon, J. & Gómez, E. (2012). Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(6), 1759–1770. [Cited on page 22.]

Salamon, J. J. (2013). *Melody Extraction from Polyphonic Music Signals.* PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain. [Cited on page 22.]

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., & Chen, X. (2016). Improved Techniques for Training GANs. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, NeurIPS 2016*, pp. 2226–2234. Curran Associates, Inc. [Cited on page 111.]

Schaeffer, P. (1952). *A la recherche d'une musique concrète.* Paris: Editions du Seuil. [Cited on page 2.]

Schedl, M., Gómez, E., & Urbano, J. (2014). Music Information Retrieval: Recent Developments and Applications. *Foundations and Trends® in Information Retrieval*, *8*(2-3), 127–261. [Cited on page 23.]

Scheirer, E. D. (1998). Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, *103*(1), 588–601. [Cited on page 21.]

Sha'ath, i. (2011). *Estimation of key in digital music recordings.* MSc thesis, Birkbeck College, University of London, London, UK. [Cited on pages 56, 150, and 151.]

Shen, Y., Gu, J., Tang, X., & Zhou, B. (2020). Interpreting the Latent Space of GANs for Semantic Face Editing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pp. 9240–9249. CVF / IEEE. [Cited on pages 89, 116, and 120.]

Shen, Y., Yang, C., Tang, X., & Zhou, B. (2022). InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(4), 2004–2018. [Cited on pages 123 and 124.]

Shen, Y. & Zhou, B. (2021). Closed-Form Factorization of Latent Semantics in GANs. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2021*, pp. 1532–1540. CVF / IEEE. [Cited on pages 89, 116, 119, and 123.]

Shepard, R. N. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, *27*(2), 125–140. [Cited on page 145.]

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. [Cited on page 85.]

Shi, Z. & Mysore, G. J. (2018). LoopMaker: Automatic creation of music loops from pre-recorded music. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18*, p. Paper 454. ACM. [Cited on pages 60 and 61.]

Smaragdis, P. (2004). Non-negative Matrix Factor Deconvolution; Extraction of Multiple Sound Sources from Monophonic Inputs. In C. G. Puntonet & A. Prieto (Eds.) *Independent Component Analysis and Blind Signal Separation: 5th International Conference, ICA 2004.*, Lecture Notes in Computer Science (LNCS), pp. 494–499. Springer. [Cited on page 60.]

Smith, J. B., Kawasaki, Y., & Goto, M. (2019). Unmixer: An Interface for Extracting and Remixing Loops. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.) *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pp. 824–831. ISMIR. [Cited on page 60.]

Smith, J. B. L. & Goto, M. (2018). Nonnegative Tensor Factorization for Source Separation of Loops in Audio. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, pp. 171–175. IEEE. [Cited on pages 57, 60, 67, 69, 71, and 72.]

Southall, C., Stables, R., & Hockman, J. (2017). Automatic Drum Transcription for Polyphonic Recordings Using Soft Attention Mechanisms and Convolutional Neural Networks. In S. Jo, Z. Duan, X. Hu, & D. Turnbull (Eds.) *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pp. 606–612. ISMIR. [Cited on page 108.]

Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. In E. Gómez, X. Hu, E. Humphrey, & E. Benetos (Eds.) *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 334–340. ISMIR. [Cited on pages 91, 92, 96, 98, 103, and 109.]

Stöter, F.-R., Uhlich, S., Liutkus, A., & Mitsufuji, Y. (2019). Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software*, *4*(41), 1667. [Cited on page 80.]

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction.* Adaptive Computation and Machine Learning series. Cambridg, MA: MIT press, 2nd edn. [Cited on pages 25 and 26.]

Tapscott, D. (1996). *The digital economy: Promise and peril in the age of networked intelligence.* New York, NY: McGraw-Hill, 1st edn. [Cited on page 5.]

Temperley, D. (1999). What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered. *Music Perception*, *17*(1), 65–100. [Cited on pages 56, 150, and 151.]

Tolstikhin, I. O., Bousquet, O., Gelly, S., & Schölkopf, B. (2018). Wasserstein Auto-Encoders. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings.* ICLR. [Cited on pages 92 and 96.]

Tymoczko, D. (2008). Set-Class Similarity, Voice Leading, and the Fourier Transform. *Journal of Music Theory*, *52*(2), 251–272. [Cited on pages 146 and 150.]

Tymoczko, D. (2010). *A geometry of music: Harmony and counterpoint in the extended common practice.* New York, NY: Oxford University Press. [Cited on page 145.]

Tymoczko, D. & Yust, J. (2019). Fourier Phase and Pitch-Class Sum. In M. Montiel, F. Gomez-Martin, & O. A. Agustín-Aquino (Eds.) *Mathematics and Computation in Music: 7th International Conference MCM 2019, Lecture Notes in Computer Science (LNCS)*, vol. 11502, pp. 46–58. Springer International Publishing. [Cited on pages 145 and 150.]

Vande Veire, L. & De Bie, T. (2018). From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass. *EURASIP Journal on Audio, Speech, and Music Processing*, *2018*(1), 13. [Cited on pages 61 and 74.]

Wang, X., Takaki, S., & Yamagishi, J. (2020). Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 402–415. [Cited on page 110.]

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1), 37–52. [Cited on page 26.]

Won, M., Chun, S., Nieto, O., & Serrc, X. (2020). Data-Driven Harmonic Filters for Audio Representation Learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020*, pp. 536–540. IEEE. [Cited on pages 62 and 63.]

Wu, F.-H. F. & Jang, J.-S. R. (2014). A supervised learning method for tempo estimation of musical audio. In *22nd Mediterranean Conference on Control and Automation*, pp. 599–604. IEEE. [Cited on page 21.]

Xu, D. & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, *2*(2), 165–193. [Cited on page 26.]

Yadati, K., Larson, M. A., Liem, C. C., & Hanjalic, A. (2014). Detecting Drops in Electronic Dance Music: Content based approaches to a socially significant music event. In H.-M. Wang, Y.-H. Yang, & J. H. Lee (Eds.) *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, pp. 143–148. ISMIR. [Cited on pages 73 and 74.]

Yang, C., Shen, Y., & Zhou, B. (2021). Semantic Hierarchy Emerges in Deep Generative Representations for Scene Synthesis. *International Journal of Computer Vision*, *129*(5), 1451–1466. [Cited on pages 89 and 116.]

Ycart, A. & Benetos, E. (2018). Polyphonic Music Sequence Transduction with Meter-Constrained LSTM Networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, pp. 386–390. IEEE. [Cited on page 144.]

Yust, J. (2017). Probing Questions About Keys: Tonal Distributions Through the DFT. In O. A. Agustín-Aquino, E. Lluis-Puebla, & M. Montiel (Eds.) *Mathematics and Computation in Music: 6th International Conference MCM 2017*, *Lecture Notes in Computer Science (LNCS)*, vol. 10527, pp. 167–179. Springer International Publishing. [Cited on pages 146 and 148.]

Yust, J. (2019). Stylistic information in pitch-class distributions. *Journal of New Music Research*, *48*(3), 217–231. [Cited on pages 145 and 148.]

Zapata, J. R., Davies, M. E. P., & Gómez, E. (2014). Multi-Feature Beat Tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(4), 816–825. [Cited on page 55.]

Zhao, Y., Po, L.-M., Cheung, K.-W., Yu, W.-Y., & Rehman, Y. A. U. (2021). SCGAN: Saliency Map-Guided Colorization With Generative Adversarial Network. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(8), 3062–3077. [Cited on page 90.]

Zhu, J.-Y., Krähenbühl, P., Shechtman, E., & Efros, A. A. (2016). Generative Visual Manipulation on the Natural Image Manifold. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.) *Computer Vision: 14th European Conference ECCV 2016, Part V*, *Lecture Notes in Computer Science (LNCS)*, vol. 9909, pp. 597–613. Springer International Publishing. [Cited on page 89.]

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision, ICCV 2017*, pp. 2242–2251. IEEE. [Cited on page 89.]

Zukowski, Z. & Carr, C. J. (2018). Generating Black Metal and Math Rock: Beyond Bach, Beethoven, and Beatles. Http://arxiv.org/abs/1811.06639. [Cited on pages 91 and 105.]

Zölzer, U. (2008). *Digital audio signal processing*. Chichester, UK: John Wiley & Sons, 2nd edn. [Cited on page 97.]

Zölzer, U. (2011). *DAFX: Digital Audio Effects*. Chichester, UK: John Wiley & Sons, 2nd edn. [Cited on pages 38 and 39.]