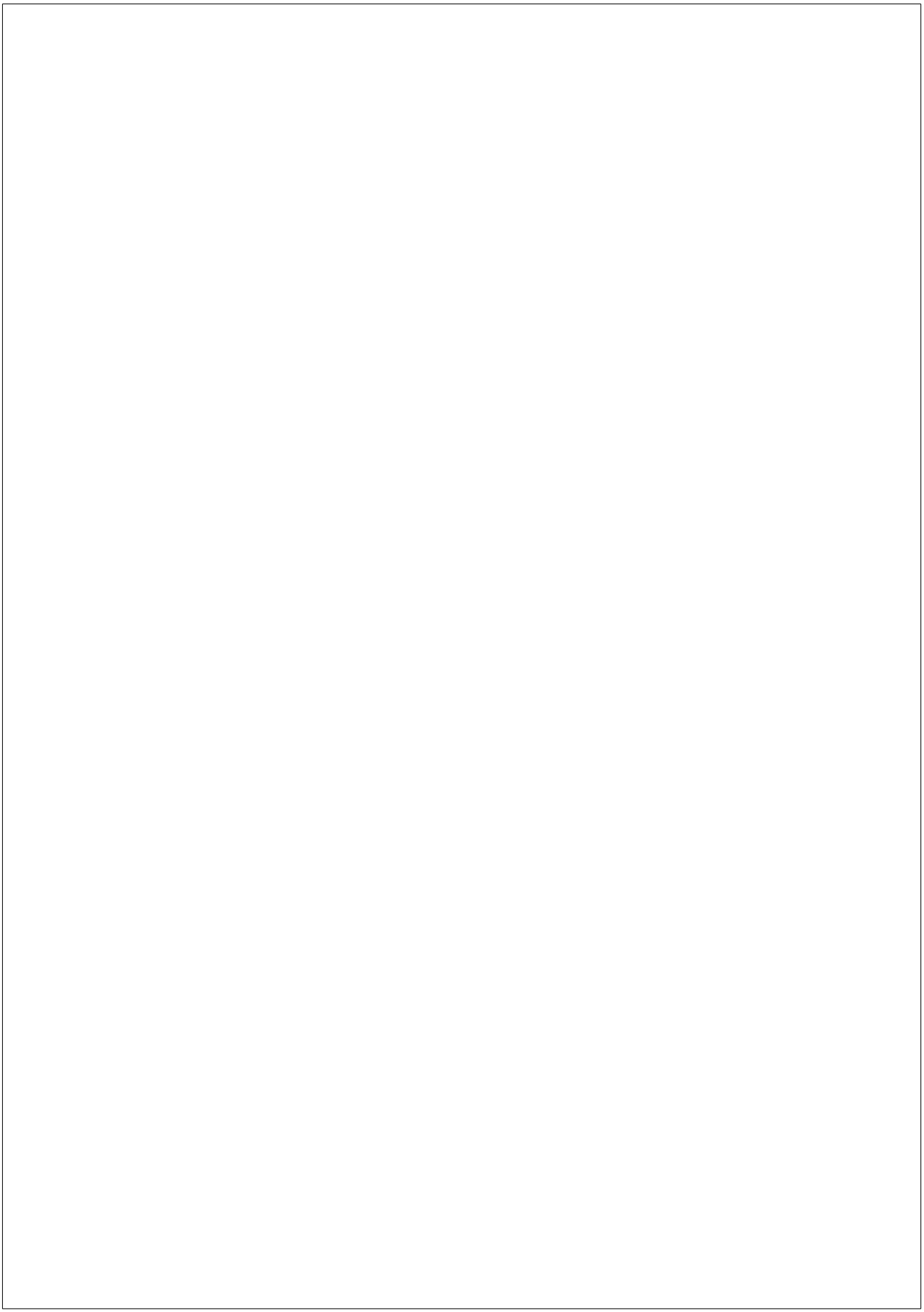
Relative Music Loudness Estimation In TV Broadcast Audio Using Deep Learning

An industrial perspective

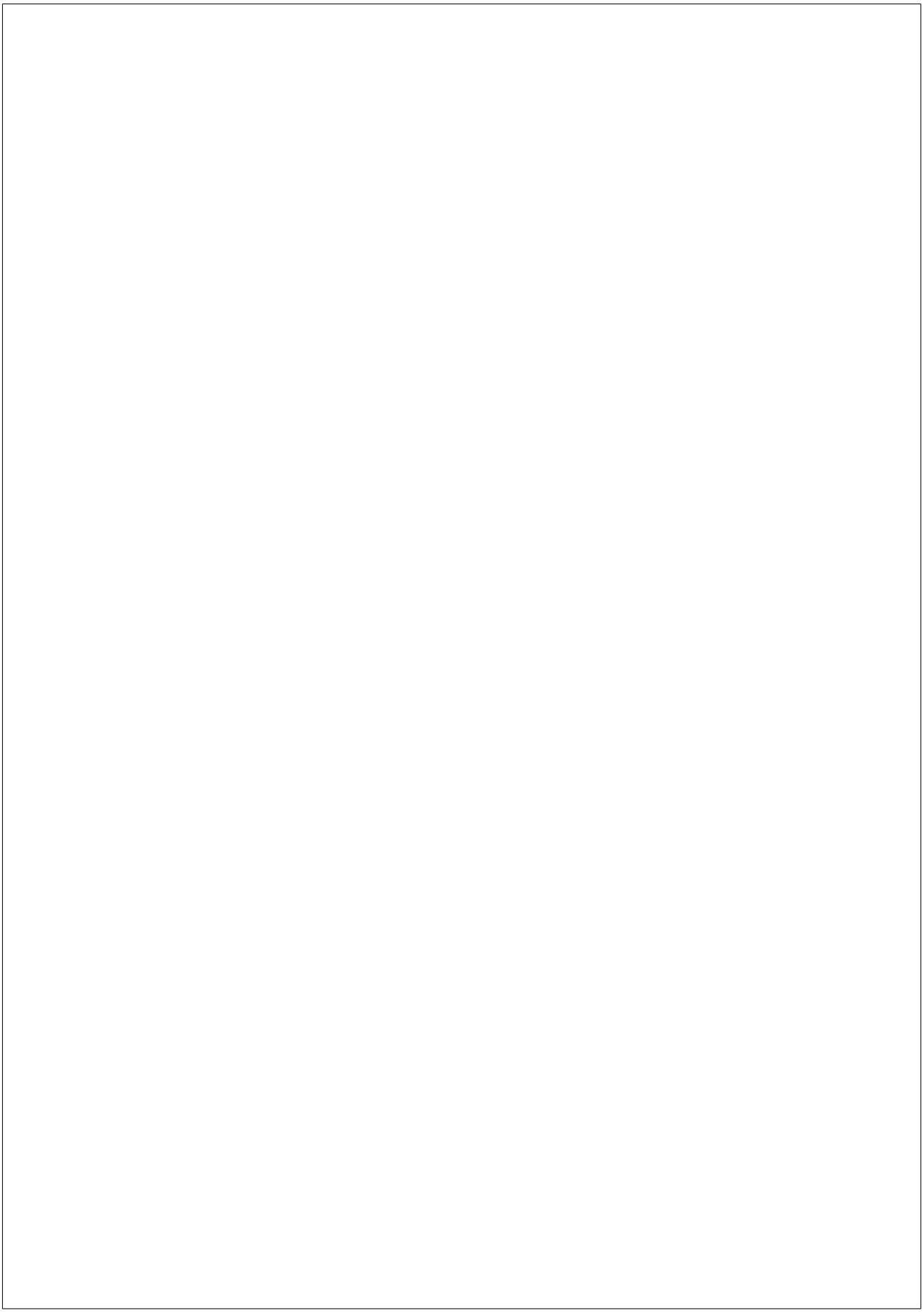
Author: Blai Meléndez Catalán

TESI DOCTORAL UPF / year 2021

THESIS SUPERVISORS
Emilio Molina, Emilia Gómez
Department: DTIC, MTG, MIR Lab

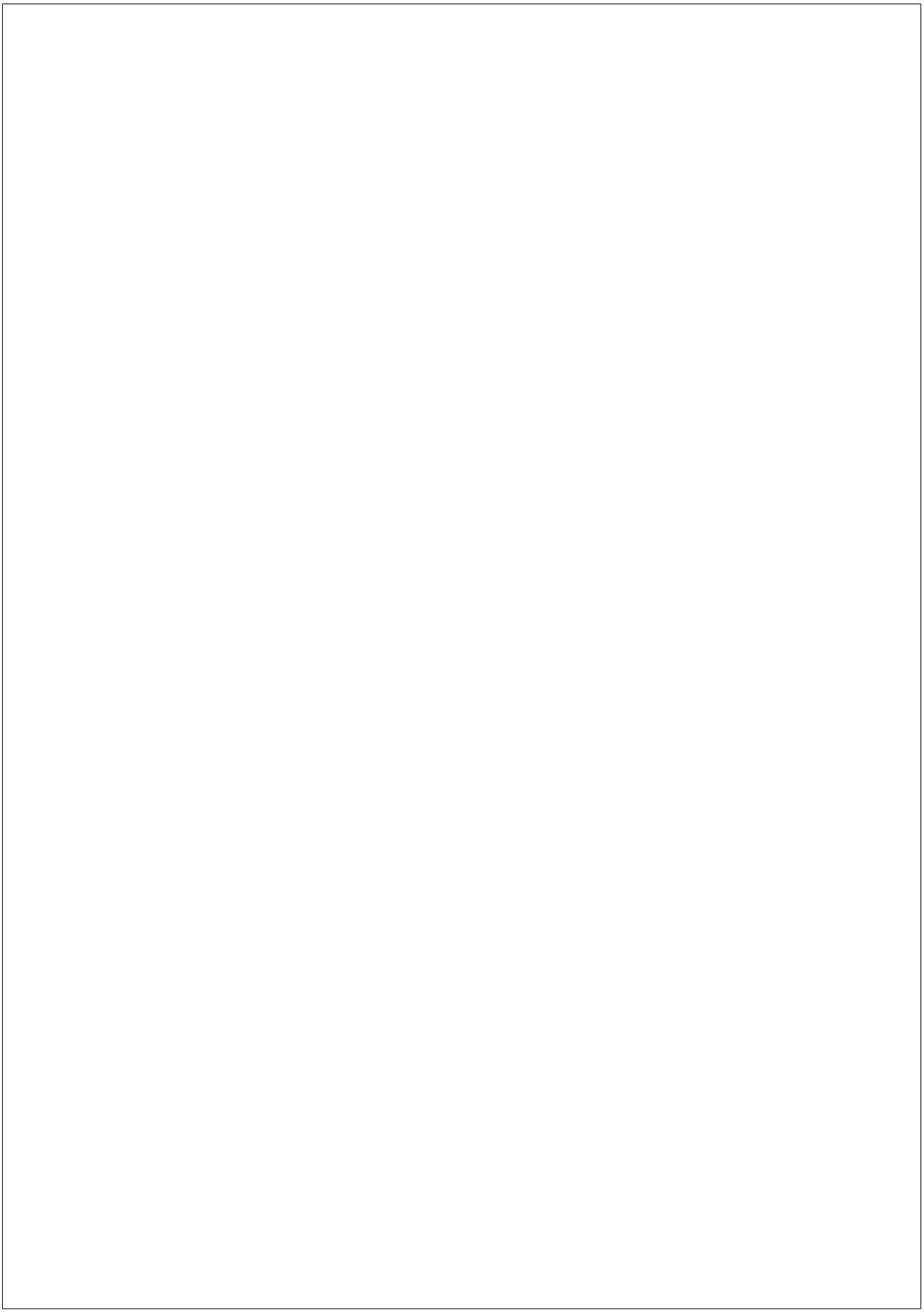


Dedico aquesta tesi doctoral a la meva família i les altres persones que em donen suport dia sí i dia també.



Acknowledgements

Many people have been involved at some point in the development of this thesis, and I am grateful for all of their contributions, because, big or small, they have been fundamental for the successful completion of this PhD thesis. First and foremost, I would like to thank my two supervisors, Emilio Molina and Emilia Gómez, for their great disposition and initiative, and the knowledge and know-how that they have poured into me. I also want to extend a special thanks to Xavier Serra for agreeing to a fateful meeting that let me into the Sound and Music Computing master, and that eventually allowed me to start working at BMAT as an intern. I am also very grateful to Johannes Lyda for believing in me and giving me the opportunity to carry out this industrial PhD and to stay at BMAT. During these four years of doctoral work, I have received advice or I have collaborated for one reason or another with: Alex Ciurana, David Doukhan, Yun Hao, Olga Slizovskaia, Pritish Chandna, Furkan Yesiler, Ana Maria Estrada, Daniel G. Camhi, Carles Antón, Cristina Garrido, Sonia Espí, Lydia Garcia, Pilar Callau, and Andy Hughes. I would like to thank them all for their help. Finally, a big thanks to my family and friends for their unconditional support.



Abstract

Under the current copyright management business model, broadcasters are taxed by the corresponding copyright management organization according to the percentage of music they broadcast, and the collected money is then distributed among the copyright holders of that music. In the specific case of TV broadcasts, whether a musical piece is played in the foreground or the background is often a relevant factor that affects the amount of money collected and distributed. In recent years, the music industry is increasingly adopting technological solutions to automatize this process. We have conducted this industrial PhD at BMAT, a company that has an active role in providing these solutions: since 2015, this company has been offering a service that currently monitors about 4300 radio stations and TV channels to automatically detect the presence of music, and to classify it as foreground or background music. We name this task relative music loudness estimation. From an industrial point of view, this thesis focuses on the improvement of the technology behind the service; and from the academic point of view, it pursues the introduction and promotion of the task in the research field of music information retrieval, and provides computational approaches to it.

The industrial and academic contributions of this thesis result from logical steps towards these goals. We first create BAT: a new open-source, web-based tool for the efficient annotation of audio events and their partial loudness in the presence of other simultaneous events. We use BAT to annotate two datasets: one private and the other public. We use the private dataset for training in the development of BMAT's new relative music loudness estimation algorithm called the Deep Music Detector. The Deep Music Detector represents the first application of deep learning within

BMAT, and provides a significant boost in performance with respect to its predecessor. The public dataset, called OpenBMAT, is released in order to foster transparent, comparable and reproducible research on the task of relative music loudness estimation. We use OpenBMAT in our proposal of a novel deep learning solution to this task based on an architecture that combines regular convolutional neural networks, and temporal convolutional networks. This architecture is able to extract robust features from a time-frequency representation of an audio file, and then model them as temporal sequences, producing state-of-the-art results with an efficient usage of the network's parameters. Finally, this thesis also offers a review of the concepts, resources and literature about tasks related to the detection of music.

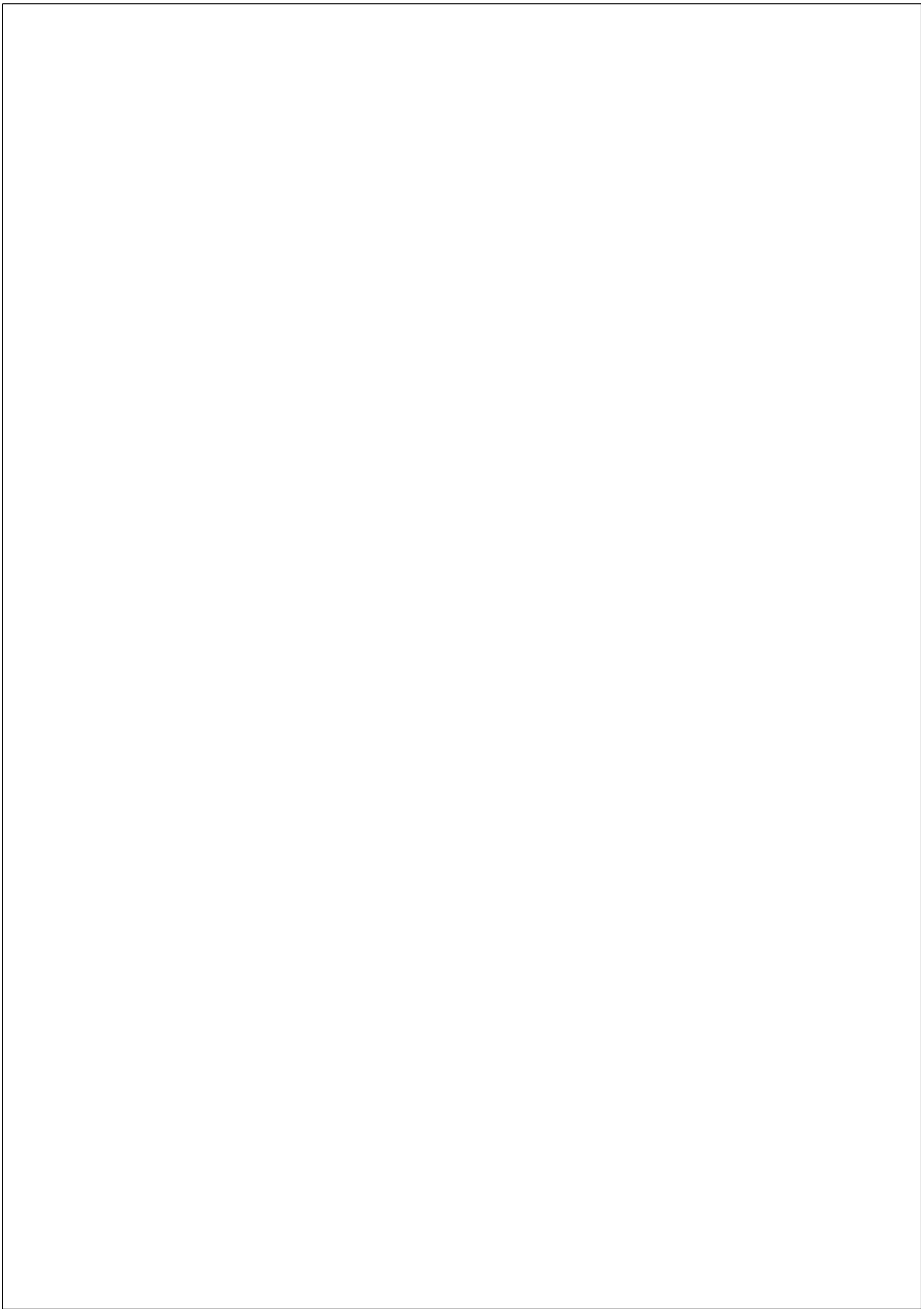
Resum

En l'actual model de negoci de la gestió de drets d'autor, les emissores paguen una certa quantitat d'impostos a l'organització de drets d'autor corresponent que depèn del percentatge de música que emeten. Els diners recaptats d'aquesta manera es distribueixen entre els propietaris dels drets d'aquesta música. En el cas específic de les emissores de televisió, el fet que la música s'emeti en primer o segon pla és sovint un factor rellevant que afecta la quantitat de diners recaptada i distribuïda. Recentment, la indústria musical està optant cada cop més per solucions tecnològiques que automatitzen aquest procés. Hem realitzat aquest doctorat industrial a BMAT, una empresa que proveeix aquest tipus de solucions. Des de 2015, aquesta empresa ofereix un servei que actualment monitora al voltant de 4300 canals de ràdio i televisió per detectar automàticament la presència de música i identificar si es troba en primer o segon pla. A aquesta tasca l'anomenem estimació del volum relatiu de la música. Des del punt de vista industrial aquesta tesi se centra en la millora de la tecnologia que hi ha darrere d'aquest servei, mentre que des del punt de vista acadèmic persegueix la introducció i promoció de la tasca en el camp de recerca del music information retrieval, i hi aporta solucions tecnològiques.

Les contribucions industrials i acadèmiques d'aquesta tesi són el resultat d'uns passos lògics, encaminats cap a la consecució aquests objectius. El primer pas és la creació de BAT: una nova eina web i de codi obert per a l'anotació d'esdeveniments acústics i del seu volum parcial. El segon pas consisteix a utilitzar BAT per anotar dos datasets: un de privat i un de públic. El dataset privat l'usem per a entrenament en el desenvolupament del Deep Music Detector, el nou algorisme d'estimació del volum relatiu de la música de BMAT. El Deep Music Detector representa la

primera aplicació d’aprenentatge profund dins de BMAT, i aporta una millora substancial del servei respecte al seu predecessor. El dataset públic, anomenat OpenBMAT, es publica per promoure una recerca transparent, comparable i reproduïble en la tasca d’estimació del volum relatiu de la música. A més a més, nosaltres l’usem en la nostra proposta d’una nova solució a aquesta tasca, que es basa en una arquitectura d’aprenentatge profund que combina les xarxes neuronals convolucionals estàndard amb les xarxes convolucionals temporals. Aquesta arquitectura permet extreure descriptors robustos a partir d’una representació temporal-freqüencial d’un fitxer d’àudio i modelar-los com a seqüència temporal. Els resultats obtinguts superen l’estat de l’art amb un ús eficient dels paràmetres de la xarxa. Finalment, aquesta tesi també ofereix una revisió dels conceptes, dels recursos i de la literatura sobre tasques relacionades amb la detecció de música.

Contents



List of Figures

List of Tables

List of abbreviations

ADAM adaptive moment. 18

BiGRU bidirectional gate recurrent unit. 52

BiLSTM bidirectional long short-term memory. 52

BMAT BMAT Licensing S.L. 6

CFA continuous frequency activation. 45

CMO copyright management organization. 3

CNN convolutional neural network. 11

DAFx digital audio effects. 107

DBN deep belief network. 58

DCASE detection and classification of acoustic scenes and events. 104

FFNN feedforward neural network. 16

GMM gaussian mixture models. 48

GRU gate recurrent unit. 52

HMM hidden markov model. 60

INA french national institute of audiovisual. 119

ISMIR international society for music information retrieval. 11

kNN k-nearest neighbors. 48

LSTM long short-term memory. 52

LU loudness unit. 4

LUFS loudness unit to full scale. 4

mcRBM mean-covariance restricted Boltzmann machines. 58

MFCC Mel-frequency cepstrum coefficients. 48

MIR music information retrieval. 1

MIREX music information retrieval evaluation exchange. 38

MLP multi-layer perceptron. 18

OpenBMAT open broadcast media audio from TVs. 10

ORF austrian national broadcasting corporation. 36

RBM restricted Boltzmann machines. 58

ReLU rectified linear units. 109

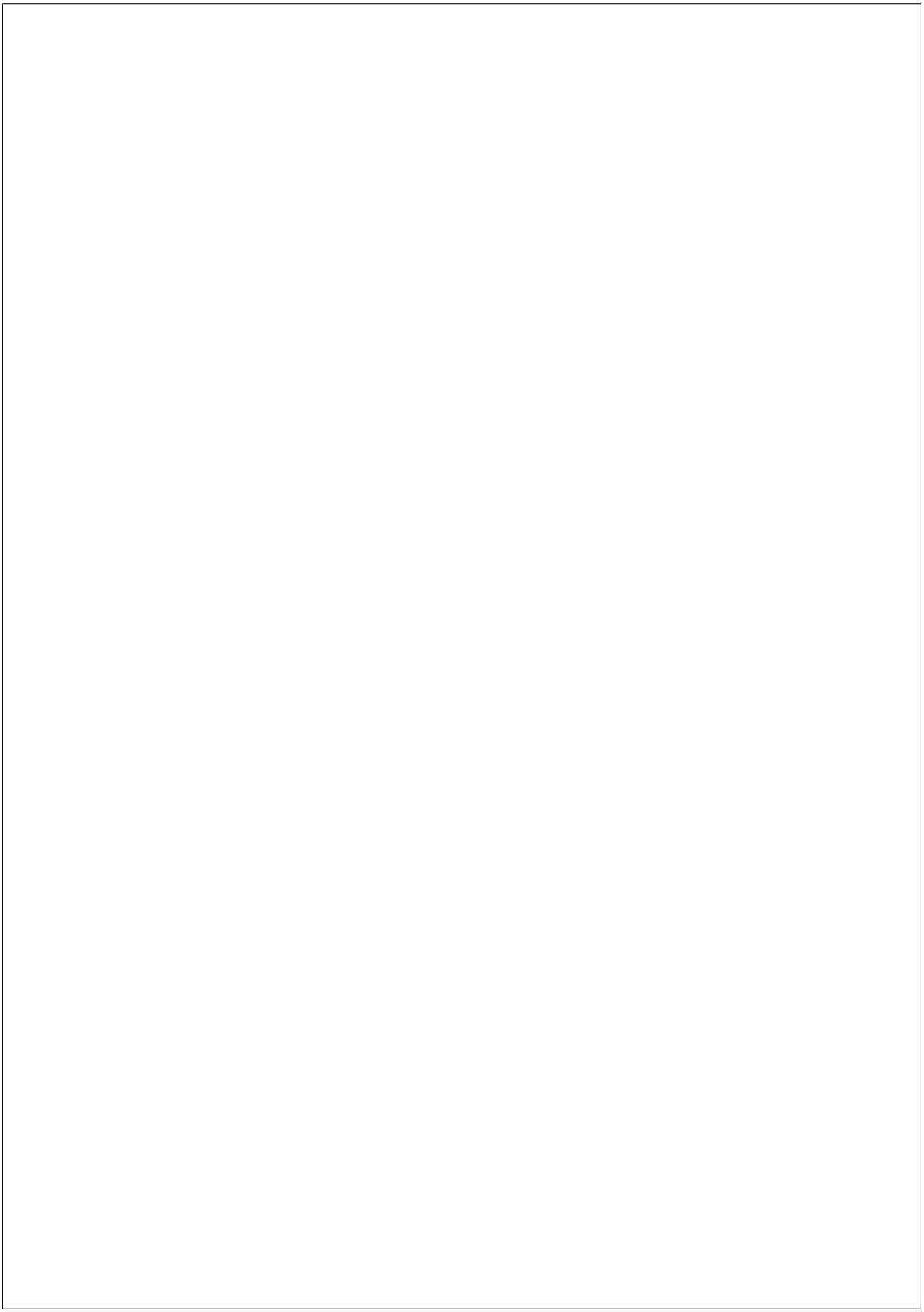
RNN recurrent neural network. 11, 51

SNR signal-to-noise ratio. 48

SVM support vector machine. 7

TCN temporal convolutional network. 11

TISMIR transactions of the international society for music information retrieval. 85



Chapter 1

INTRODUCTION

1.1 Motivation

This PhD thesis deals with the detection of music in broadcast media audio and the estimation of its loudness in relation to simultaneous non-music sounds. This exceeds the task of music detection, which has been a topic of interest in the music information retrieval (MIR) field for more than a decade, and acts as a stepping stone for a new line of research. To motivate the need to start this new line of research, we first need to describe how music is used in broadcast TV audio, and the applications and limitations of the music detection task.

Music can play many roles in TV: it may be the main focus of attention or just a means to create a certain atmosphere, it may be linked to a certain content or to a program section, etc. In addition, it can appear either isolated or mixed with non-music sounds. The variety of these non-music sounds is large, and the range of the music loudness relative to them is broad. To illustrate this variety of situations, we provide the following

three examples:

- In music videos and live music performances, music is the most important part of the content; however, while in music videos music generally appears isolated, in live music performances it appears mixed with sounds from the audience such as cheering and applause.

- In speech-predominant programs such as news broadcasts or talk shows, music is typically used in the background, sometimes at a very low volume, to create tension and to separate different sections of the program.

- In movies, music is used to build a certain atmosphere and to recall certain emotions, but it can do so both as foreground and background music, and in isolation or mixed with non-music sounds of any type.

Music detection[1] refers to the task of finding time intervals containing music in an audio file, or what is the same, segmenting the audio file in time intervals of two classes: *Music* and *No Music*. With *No Music* we refer to the absence of music. The two main applications of music detection algorithms are (1) the automatic indexing and retrieving of information based on its audio content, and (2) the monitoring of music for copyright management (Zhu et al., 2006; Seyerlehner et al., 2007; Izumitani et al., 2008; Giannakopoulos et al., 2008). Additionally, the detection of music

[1] https://www.music-ir.org/mirex/wiki/2019:Music_Detection

can be applied as a preprocessing step to improve the performance of algorithms designed for other purposes such as audio fingerprinting (Gfeller et al., 2017).

The present work is framed in the context of the industrial copyright management application. Under the current copyright management business model, broadcasters are taxed based on the percentage of music they broadcast. Typically, broadcasters report the broadcast music to a copyright management organization (CMO) and these reports are used to define their taxes, which will later be distributed among the copyright holders of this music.

The reason behind the need to estimate the music's loudness in relation to simultaneous non-music sounds is that, in the case of TV broadcast audio, CMOs consider whether the broadcast music is used in the foreground or the background as one of several relevant factors for the distribution of copyright royalties.[2,3,4,5] Other relevant factors are, for instance, the duration of the music or the audience measurement. In this scenario, music detection algorithms fall short, and the need for the definition of a new task that takes the relative loudness of music into consideration becomes apparent. We name this task relative music loudness estimation.

Scharf (1978) defined loudness as "the perceived or subjective intensity of a sound". It is also defined by the American National Standards Institute as "that attribute of auditory sensation in terms of which sounds

[2] https://createurs-editeurs.sacem.fr/actuimg/fr/live/v4/Createurs-Editeurs/docs_et_brochures/Repartition/sacem_regles_repartition2017.pdf

[3] https://www.bmi.com/creators/royalty_print#id-533116

[4] https://www.ascap.com/help/music-business-101/music-money-success-movies

[5] https://stockmusic.net/blog/the-ultimate-guide-to-cue-sheets/

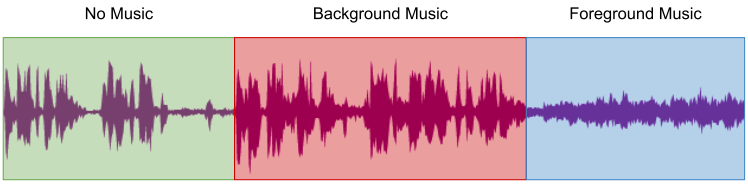


Figure 1.1: Example of the division of an audio file in time intervals of the classes of the relative music loudness estimation task.

can be ordered on a scale extending from quiet to loud” (American National Standards Institute, 1973). The unit to measure the loudness used in media broadcasting is the loudness unit to full scale (LUFS), introduced in EBU R128[6]. This measure is used to normalize broadcast audio at a standard level of -23 LUFS. A unit of LUFS is called loudness unit (LU) and is equal to 1 decibel.

Cartwright et al. (2018) used the concept of relative loudness to refer to a real value representing the loudness of an event with respect to other simultaneous events. We use a simplification of this concept that only specifies if an event is in the foreground or the background with respect to these simultaneous events. Moreover, we exclusively focus on the relative loudness of music events: we define the relative music loudness estimation task as a segmentation task that consists of dividing audio into time intervals of *Foreground Music*, *Background Music* and *No Music*. Throughout this thesis, we generally spell out all the words when mentioning task names; however, in some figures and tables, and in particular sections, we use RMLE as an acronym for relative music loudness

[6] `https://tech.ebu.ch/docs/r/r128.pdf`

estimation, and MD as an acronym for music detection.

Scharf (1964) and also Zwicker and Fastl (1999) defined the loudness of a target sound in the presence of a simultaneous partially masking sound as the partial loudness of the target sound. In this thesis, we interpret that the partial loudness pl_e of an event e in the presence of other simultaneous events, is also an approximation of its contribution to the total loudness L_E of the set of simultaneous events E, where $e \in E$, as shown in eq. 1.1.

$$L_E \approx \sum_{e \in E} pl_e \tag{1.1}$$

In this thesis, we only use this interpretation of the concept of partial loudness as a guide for the annotators to generate manual annotations that we can later map into the relative music loudness classes. In Section 3.2.2, we describe how partial loudness is used during the annotation process, and in Section 4.1.2, we show the heuristic rules that we use to map the partial loudness annotations to a relative loudness ground truth. In general, we consider that music is in the foreground only if it has the highest partial loudness among all the simultaneous events; otherwise, we consider that it is in the background.

1.2 BMAT as a company

Currently, there are still CMOs that manually check the reports from the broadcasters using a sampling of the total broadcast time. This procedure is typically inaccurate leading to mistakes that result in an unfair distribution of royalties. As the music business community is becoming aware

of the power of machine learning and its applications to MIR, the interest in systems for the automatic analysis of broadcast material is increasing. In the context of the industrial PhD program[7], we carry out our doctoral work at BMAT Licensing S.L (BMAT)[8], a company that has been working on this automation since its foundation in 2005.

Using its audio fingerprinting technology, BMAT monitors music across televisions, radios, venues and digital services globally, and reports the metadata that describes who owns the rights to each identified track to any party involved in the value cycle: creators, producers, publishers, CMOs, broadcasters, digital service providers, and clubs. Its database contains more than 72 million audio fingerprints of commercial, production and commissioned music via direct partnerships with more than 120,000 content owners. BMAT continuously monitors around 5,000 radio stations and 1,500 television channels in 134 countries, and over 1,000 clubs on five continents. On top of that, it processes 100 million sales from Youtube, Spotify, Apple Music, Amazon and 40 more digital service providers every hour.

Relative music loudness estimation is another service that BMAT has offered since 2015, continuously monitoring around 4300 radio stations and TV channels. As a service, it is essential for it to have actual use cases and potential customers. BMAT uses the relative music loudness estimation service in four different situations:

- To calculate the amount of foreground and background music played in each channel for tax-paying purposes. This is the most common use case and it applies to CMOs and broadcasters. Broad-

[7] http://doctoratsindustrials.gencat.cat/
[8] https://www.bmat.com/

casters are taxed by the CMOs according to this information, so both of them are interested in it.

- To detect music played in any of the monitored channels that BMAT does still not have in its audio fingerprinting database. BMAT analyzes the parts of the audio where music is detected but not identified and, if a song is found, it is ingested into a particular pool for this kind of content. This is an internal use case that helps improving BMAT's audio fingerprinting service.

- To mark parts of the audio that have a low probability of containing music, so that they do not have to be analyzed by the audio fingerprinting technology. This is another internal use case that helps to increase the efficiency of BMAT's audio fingerprinting service.

- To analyze the catalog of other companies to assess the percentage of non-music content such as audio-books, podcasts, sound effects, etc. For this use case, the algorithm does not divide a track into time intervals of different class, but assigns a single label to it: either *Music* or *No Music*. This is done through an extra layer that feeds on the relative music loudness information to produce a label.

1.3 Goals and contributions

In 2016, prior to the start of this doctoral work, BMAT's relative music loudness estimation algorithm relied on a support vector machine (SVM) algorithm (Cortes and Vapnik, 1995) and a set of features extracted using the Essentia library (Bogdanov et al., 2013). The relevance of this service was a great motivation for BMAT to promote the necessary research and

development to improve the technology behind it. The main goals of this thesis are:

Goal 1: The development of state-of-the-art computational approaches to the task of relative music loudness estimation.

From BMAT's point of view, this must result in a new production-ready algorithm. From the academic point of view, this thesis also pursues:

Goal 2: The introduction and promotion of the relative music loudness estimation task in the research field of MIR.

An industrial PhD requires industry and academia to work together through a collaboration agreement that must be of benefit to the two parts. This means that achieving these goals should translate into useful contributions to both BMAT and the research community. This thesis produces the following contributions while advancing towards the attainment of its goals:

- **Introduction of the relative music loudness estimation task as a new MIR task.** Given the importance of differentiating between foreground and background music in the negotiations between CMOs and TV broadcasters, we propose the task of relative music loudness estimation as a new task in the research field of MIR. We expect this to promote technological advances that BMAT can leverage in the future.

- **Review of the literature and the available resources related to the detection of music.** The task of relative music loudness estimation is a new task that we define in this thesis; thus, there is no previous research about it. However, its similarity to the task of music detection allows us to take advantage of much of the literature and the available resources related or involving this task for our own purposes. This thesis offers a thorough review of this literature and these resources in Chapter 2.

- **Development of an audio-events annotation tool with a focus on the annotation of an event's partial loudness.** We present BAT: an open-source, web-based, annotation tool that is specific for the annotation of audio events. We design BAT to allow for the efficient annotation of an event's partial loudness in the presence of other simultaneous events. Furthermore, it is easy to use and to deploy in servers, and it is sufficiently versatile as to meet the annotation needs of other researchers in the MIR community. We provide a thorough description of BAT in Chapter 3.

- **Creation and annotation of a private dataset for the task of relative music loudness estimation.** We sample BMAT's private database to compile a dataset comprising approximately 44 hours of audio from TV channels and radio stations from all over the world distributed in 1322 two-minutes files. It includes the manual annotations of a single annotator that used a taxonomy of four classes: *Music*, *Speech*, *Sound effects* and *Audience*. The annotations also include information about the partial loudness of these types of audio events when they overlap. We provide an extended description

of this dataset in Section 4.1.

- **Creation and annotation of a public dataset for the tasks of relative music loudness estimation and music detection.** We sample BMAT's private database again to create open broadcast media audio from TVs (OpenBMAT), a public dataset containing 27 hours of audio divided in 1647 one-minute audio files that come from well-known TV channel in France, Germany, Spain and the United Kingdom. These audio files belong to one of eight program types: children programs, documentaries, entertainment programs, music programs, news broadcasts, series and films, sport programs, and talk shows. OpenBMAT includes manual annotations about the presence of music and its relative loudness with respect to simultaneous non-music sounds. We describe this dataset in Section 4.2.

- **Development of BMAT's new relative music loudness estimation algorithm.** Using the private dataset introduced above, we train the Deep Music Detector.[9] This is the relative music loudness estimation algorithm that BMAT is currently using in production, and it gives significantly better performance than its SVM predecessor. It also represents BMAT's first attempt at using deep learning.[10] We explain the details of this in Section 5.1.

- **Development of state-of-the-art deep learning computational approaches for the estimation of music's relative loudness.** We propose two deep learning architectures for this task: the isolated

[9] https://www.bmat.com/blog/2019/02/21/music-detection/

[10] https://www.bmat.com/blog/2018/03/22/keep-learning-deep-learning/

temporal convolutional network (TCN), and a TCN in combination with a convolutional neural network (CNN) front-end. The second architecture represents a novel type of network, which we call CNN-TCN, inspired by the usage of a CNN in concatenation with a recurrent neural network (RNN) to improve the performance of the RNN (Lemaire and Holzapfel, 2019; de Benito-Gorron et al., 2019). We detail these architectures in Section 5.2. Note that, in this thesis, we use CNN to refer to the specific case of a two-dimensional CNN. Otherwise, we write 1D-CNN.

- **Organization of the MIREX competition as captain of the tasks of relative music loudness estimation, music detection, and speech detection.** MIREX[11] is an international annual evaluation campaign for MIR algorithms, coupled to the international society for music information retrieval (ISMIR)[12] conference. In 2018 and 2019, we organized the evaluation of these tasks providing evaluation datasets, running the submitted algorithms, and reporting their results. In addition, we also participated as authors with the Deep Music Detector obtaining outstanding results. We provide greater detail in Section 5.1.5.1.

1.4 Structure of the thesis

So far, we have described the industrial context in which this thesis takes place, as well as the motivation behind it. We have also stated its goals,

[11] `\footnote{\url{https://www.music-ir.org/mirex/wiki/MIREX_HOME}}`

[12] `https://www.ismir.net/`

and the contributions to BMAT and academia, that lead us to the attainment of these goals. In Chapter 2, we present several technical concepts and digital resources that are relevant for the thesis, and also a literature review, where we provide details about the evolution of the research on tasks that involve the detection of music. Finally, we present the conclusions that we extracted from all the information gathered in this chapter. These conclusions define the path forward that we have followed to achieve the goals of this thesis. The path forward includes three steps: the creation of an audio-events annotation tool, the annotation of datasets that are suitable to the task of relative music loudness estimation, and the development of computational approaches for this task. We devote the following three chapters to describing the work that we have carried out and the contributions that we have made in following this path.

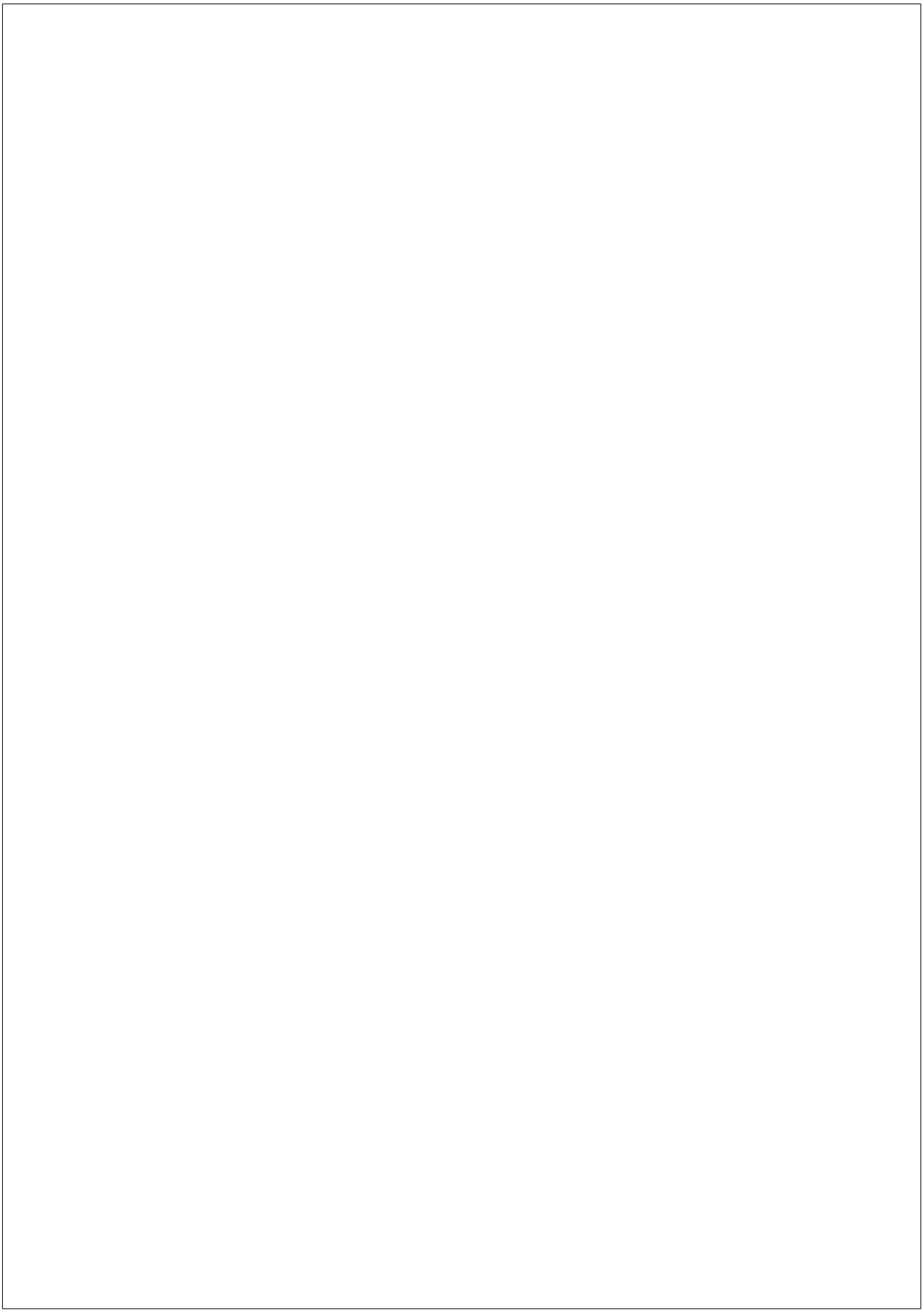
Chapter 3 is about the development of BAT, the audio-events annotation tool. This tool allows for an efficient annotation of an event's partial loudness in the presence of other simultaneous audio events. This makes it suitable to produce annotations for the relative music loudness estimation task. We first set out the reasons behind the decision to create our own tool and then specify the design requirements. After this, we provide a detailed explanation of the development of the tool, and finally, we conduct an experiment to verify that the partial loudness annotation system is reliable.

Chapter 4 presents a description of the audio content, the annotation process, and the resulting annotations of the two datasets that we annotate for the relative music loudness estimation and the music detection tasks using BAT. The audio included in these datasets is extracted from BMAT's private database. One of these datasets remains private and be-

comes the training dataset for the Deep Music Detector. We release the other dataset, called OpenBMAT, as a public dataset with the intention of promoting open research in the field of MIR.

In Chapter 5, we describe the computational approaches to the task of relative music loudness estimation that we have produced throughout this thesis. The first approach, developed using the private dataset, is the Deep Music Detector. We provide an overview of this algorithm and explain the details of its integration into BMAT. The second approach, trained and evaluated using OpenBMAT, is an experimental approach based on advanced deep learning architectures such as TCNs, and also their combination with a CNN front-end, which results in the novel CNN-TCN architecture.

Finally, in Chapter 6, we summarize all the contributions of the thesis and confirm that we have accomplished its goals. We also provide a list of the publicly available outcomes of our doctoral work. Finally, we devote the last section of the chapter to an overview of the future perspective of our research.



Chapter 2

SCIENTIFIC BACKGROUND

As explained in Section 1.1, this doctoral work proposes a new task consisting of the estimation of music's loudness relative to simultaneous non-music sounds. As a new line of research, there are no published approaches specifically addressing this; however, the foundations of this new research topic can be found in various related tasks. These are the tasks that include the detection of music either as a binary-class task, or in combination with the detection of speech, and occasionally also other types of non-music sounds, in multi-class segmentation or detection tasks. Throughout this document, we call these tasks music detection related tasks. Clearly, relative music loudness estimation is closely related to the detection of music; basically, it includes music detection. The fundamental difference between the two is that relative music loudness estimation also takes into consideration the presence of any kind of non-music sound, which results in the emergence of the *Foreground Music* and *Background Music* classes.

This chapter has several purposes: in Section 2.1, we introduce a se-

ries of concepts that are relevant to the thesis. These concepts include, first, a formal description of the deep learning architectures that we use in our approaches to the task of relative music loudness estimation (see Chapter 5); and second, the explanation and the mathematical expressions of the evaluation metrics that are typically used in the music detection related tasks. In Section 2.2, we describe and analyze both the public and private datasets that are available for these tasks, and we provide an overview of the audio annotation tools that can be used to annotate them. Still focusing on the music detection related tasks, in Section 2.3, we carry out a literature review, providing a detailed description of previous approaches. Throughout these sections, we make observations and extract conclusions about the key points that this thesis addresses. We summarize them in Section 2.4.

2.1 Relevant concepts

In this section, we first formally define the deep learning architectural background that we later use, in Chapter 5, to create our own approach to relative music loudness estimation. Then, we introduce an intuitive description and the mathematical expressions of the most common evaluation metrics in music detection related tasks.

2.1.1 Feedforward neural networks

A feedforward neural network (FFNN) is a type of machine learning model based on the interconnection of a series of nodes called *neurons* that are sorted in a layer-wise manner. Each neuron is able to receive an input stimulus and transform it to produce and output. In a FFNN, this

flow of information is only allowed to move forward, starting at the input layer and finishing at the output layer. The goal of the model is to approximate a function f^* through the interaction of these neurons, where f^* transforms an input $\mathbf{x}$ into the desired output $\mathbf{y}$ as shown in eq. 2.1.

$$\mathbf{y} = f^*(\mathbf{x}) \tag{2.1}$$

The output $\hat{\mathbf{y}}$ of the approximated function f depends on the input $\mathbf{x}$ and the internal network parameters θ, as shown in eq. 2.2.

$$\hat{\mathbf{y}} = f(\mathbf{x}, \theta) \tag{2.2}$$

The difference between $\mathbf{y}$ and $\hat{\mathbf{y}}$ for an input $\mathbf{x}$ can be translated into a scalar $loss$ by means of a loss function f_{loss}, as shown in eq. 2.3.

$$loss = f_{loss}(\mathbf{y}, \hat{\mathbf{y}}) = f_{loss}(\mathbf{y}, f(\mathbf{x}, \theta)) \tag{2.3}$$

When training a network, the objective is to minimize $loss$. Given that the input $\mathbf{x}$ and its desired output $\mathbf{y}$ are known variables that are extracted from a dataset, the only way to minimize $loss$ is to find the optimal network parameters θ. There are many methods to optimize these parameters, but most of them rely on the computation of the gradient ($\nabla_\theta f_{loss}(\theta)$) of f_{loss} with respect to the current network parameters θ using the backpropagation algorithm (Rumelhart et al., 1986). The opposite of this gradient indicates the direction in which a change in θ minimizes $loss$ the most. The simplest of these methods, shown in eq. 2.4, is called gradient descent, or steepest descent.

$$\theta \longleftarrow \theta - lr * \nabla_\theta f_{loss}(\theta) \tag{2.4}$$

This method updates θ for each pair $(\mathbf{x}, \mathbf{y})$ in a dataset by subtracting a scaled version of $\nabla_\theta f_{loss}(\theta)$ from θ. The scaling factor lr is called the learning rate, and it sets the pace at which θ is changed. Other well-known methods are the stochastic gradient descent (Robbins and Monro, 1951; Kiefer et al., 1952) and the adaptive moment (ADAM) (Kingma and Ba, 2014).

The type of neural interconnections, the transformations that neurons apply, and the way they are sorted define the type of FFNN that we obtain. In this section, we describe three general types of FFNN: the multi-layer perceptron (MLP), the CNN, and the TCN.

2.1.1.1 Multi-layer perceptron

The most basic neural network is the multi-layer perceptron. We show an example of this type of network in Fig. 2.1. It consists of at least an input layer, a hidden layer, and an output layer. All layers contain a set of nodes called neurons except for the input layer, which contains the input values instead. Every non-input layer is fully-connected to the previous one, i.e., each one of its neurons is connected to all the neurons of the previous layer, or to all the values of the input in the case of the first hidden layer.

Eq. 2.5 formally describes the output $\mathbf{h}^{(1)} \in \mathbb{R}^i$ of the first hidden layer of neurons given an input $\mathbf{x}$, where i is the number of neurons of this layer. Eq. 2.6 shows the generalization of the previous equation for the output $\mathbf{h}^{(l)} \in \mathbb{R}^m$ of the lth hidden layer, which contains m neurons. This output depends on the output $\mathbf{h}^{(l-1)} \in \mathbb{R}^n$ of the previous hidden layer, which is formed by n neurons.

$$\mathbf{h}^{(1)} = \sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \tag{2.5}$$

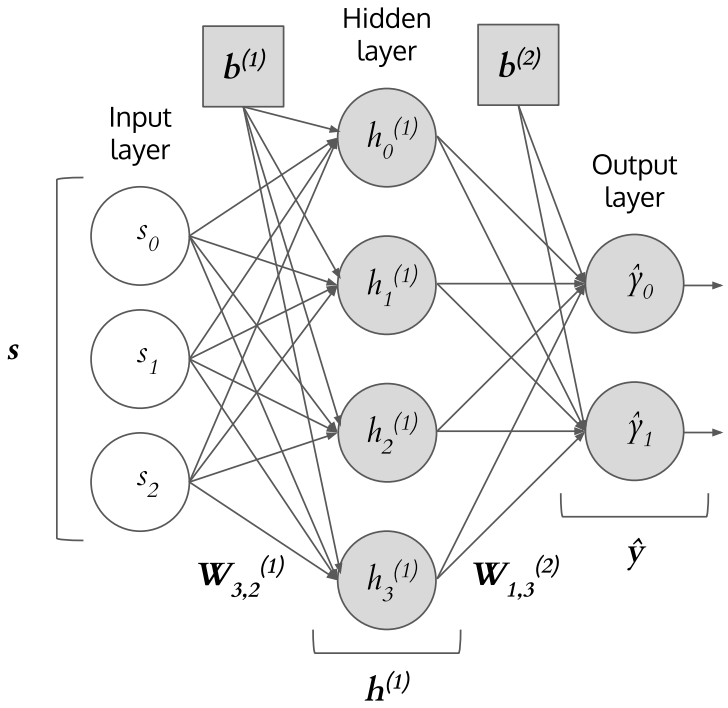


Figure 2.1: Example of an MLP with the minimum number $L = 1$ of hidden layers.

$$\mathbf{h}^{(l)} = \sigma^{(l)}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.6}$$

$\mathbf{W}^{(l)} \in \mathbb{R}^{m \times n}$ is a matrix of weights, represented in Fig. 2.1 by the arrows between two layers, $\mathbf{b} \in \mathbb{R}^m$ is a vector of biases, and σ denotes a usually non-linear function called activation function. Eq. 2.7 shows the specific case of the output layer of an MLP with $L - 1$ hidden layers, where $\hat{\mathbf{y}}$ is the prediction of the MLP.

$$\hat{\mathbf{y}} = \sigma^{(L)}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}) \tag{2.7}$$

2.1.1.2 Convolutional neural networks

Convolutional neural networks are a specific and more advanced type of neural networks where at least one layer uses the convolution of its input with a set of filters, instead of the matrix multiplication of input and weights of the MLP. The output of a 1D-convolution of an input $\mathbf{x}$ with a filter $\mathbf{k}$ of length m is

$$(\mathbf{x} * \mathbf{k})[i] = \sum_m \mathbf{x}[m]\mathbf{k}[i-m] \tag{2.8}$$

and in the case of a 2D-convolution of an input $\mathbf{X}$ with a filter $\mathbf{K}$ of size $m \times n$

$$(\mathbf{X} * \mathbf{K})[i,j] = \sum_m \sum_n \mathbf{X}[m,n]\mathbf{K}[i-m, j-n] \tag{2.9}$$

If we assume that filters have a shorter length (1D) or smaller size (2D) than the input, the output of the ith neuron in the lth convolutional layer will no longer be a scalar but a feature vector $\mathbf{h}_i^{(l)}$, in the case of 1D-convolutional layers, or a feature map $\mathbf{H}_i^{(l)}$, in the case of 2D-convolutional layers. The output for the 1D-convolutional case is formally described in eq. 2.10.

$$\mathbf{h}_i^{(l)} = \sigma((\sum_{j=0}^{J-1} \mathbf{W}_{i,j}^{(l)} * \mathbf{h}_j^{(l-1)}) + \mathbf{b}^{(l)}) \tag{2.10}$$

$\mathbf{W}_{i,j}^{(l)} \in \mathbb{R}^n$ includes the weights of the 1D-filter of length n that con-

nects the jth neuron out of the J neurons in the previous layer with the ith neuron out of the I neurons in the current layer. $\mathbf{b}^{(l)} \in \mathbb{R}^n$ is the bias vector. Eq. 2.11 extends eq. 2.10 to two dimensions.

$$\mathbf{H}_i^{(l)} = \sigma((\sum_{j=0}^{J-1} \mathbf{W}_{i,j}^{(l)} * \mathbf{H}_j^{(l-1)}) + \mathbf{B}^{(l)}) \tag{2.11}$$

$\mathbf{W}_{i,j}^{(l)} \in \mathbb{R}^{m \times n}$ includes the weights of the 2D-filter of size $m \times n$ that connects the jth neuron out of the J neurons in the previous layer with the ith neuron out of the I neurons in the current layer. $\mathbf{B}^{(l)} \in \mathbb{R}^{m \times n}$ is the bias matrix. Regardless of the dimensions of the input and the filters, these equations show that each neuron in a convolutional layer l contains J filters, and that the jth filter convolves with the output of the corresponding jth neuron of the previous layer.

It is very typical for convolutional layers to be followed by a pooling layer that helps to make the network invariant to small shifts of the input (Goodfellow et al., 2016). Max-pooling (Zhou and Chellappa, 1988) is the most common type of pooling and consists of replacing the output value of a layer at a certain location with the maximum of the nearby output values including itself.

A neuron in a CNN needs all the information in the network's input to entirely produce its output feature vector or map. However, this neuron does not use the whole network's input to produce each value of its output feature vector or map, but only a portion of it, as shown in Fig. 2.2. This portion is called the receptive field of that neuron. The receptive field of the neurons in the first convolutional layer is equal to the length or size of its filters, but the receptive field increases for the neurons of subsequent layers as the contributions from neurons of previous layers accumulate.

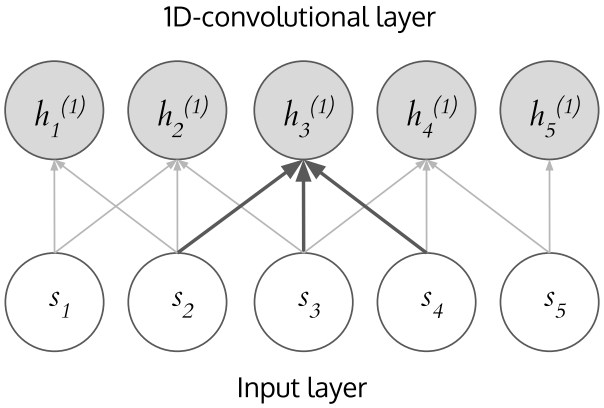


Figure 2.2: Example of a receptive field in a CNN. Neuron 3 only receives information from a portion of the input $\mathbf{s}$ to produce a value of the output feature vector $\mathbf{h}_3^{(1)}$.

2.1.1.3 Temporal convolutional networks

Temporal Convolutional Networks are 1D-CNNs formed by 1D-convolutional layers that are able to read 2D-inputs as long as one of the axis represents time. This is done by interpreting a 2D-input as a set of 1D-temporal sequences, and applying the filters of the first 1D-convolutional layer to them. In this way, TCNs process sequential data and model temporal evolution.

Typically, these 1D-convolutional layers are arranged in blocks that stack upon each other. These blocks may include more than one 1D-convolutional layer, and they usually contain residual connections (He et al., 2016; Lemaire and Holzapfel, 2019; Bai et al., 2018). We call a

block with residual connections a residual block. A residual block, as defined by He et al. (2016), applies a certain function ϕ to the output $\mathbf{H}^{(r-1)}$ of the previous block that depends on the weights $\mathbf{W}^{(r)}$ and biases $\mathbf{b}^{(r)}$ of the L layers contained in the current residual block r. The output of this function is then added back to the input and passed to an activation function to obtain the output $\mathbf{H}^{(r)}$ of the current residual block. In this way, the layers inside the residual block learn modifications to the input instead of a complete transformation. This has proven to ease their optimization (He et al., 2016). Eq. 2.12 presents the formal definition of a residual block.

$$\mathbf{H}^{(r)} = \sigma(\phi(\mathbf{H}^{(r-1)}, \mathbf{W}^{(r)}, \mathbf{b}^{(r)}) + \mathbf{H}^{(r-1)}) \tag{2.12}$$

Relating this expression with the notation of the previous sections, we have

$$\begin{aligned} \mathbf{W}^{(r)} &= \{\mathbf{W}^{(l)}\} \ \forall \ l \in \{0 \dots L-1\} \\ \mathbf{b}^{(r)} &= \{\mathbf{b}^{(l)}\} \ \forall \ l \in \{0 \dots L-1\} \\ \mathbf{H}^{(r)} &= \{\mathbf{h}_i^{(r)}\} \ \forall \ i \in \{0 \dots I-1\} \end{aligned} \tag{2.13}$$

where $\mathbf{h}_i^{(r)}$ is the ith feature vector out of the I feature vectors at the output of block r. If we consider the first residual block and a 2D-input $\mathbf{X}$, then the last expression becomes

$$\mathbf{H}^{(r)} = \sigma(\phi(\mathbf{X}, \mathbf{W}^{(r)}, \mathbf{b}^{(r)}) + \mathbf{X}) \tag{2.14}$$

It is also common for the 1D-convolutional layers in subsequent residual blocks to have a higher dilation rate (Lea et al., 2017; Bai et al., 2018).

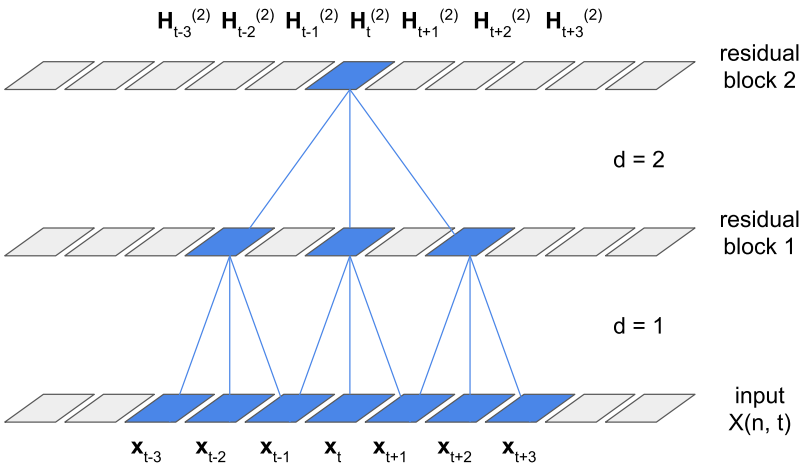


Figure 2.3: Simplified representation of a TCN. The architecture includes two residual blocks with dilations $\mathbf{d} = [1, 2]$ and non-causal filters of lengths $\mathbf{z} = [3, 3]$. The receptive field at the neurons of the second residual block is equal to 7 time-frames.

With a dilation rate d, the filters in a 1D-convolutional layer are expanded by adding $d - 1$ zeros between its weights. This greatly increases the receptive field of the network at its output, and avoids some redundancy as a smaller part of the information that flows through the network is simultaneously used by different neurons. The receptive field can consider only past and present information or also include future data depending on whether the 1D-convolutional filters are causal or non-causal (Lemaire and Holzapfel, 2019), respectively. Eq. 2.15 shows how to

compute the receptive field RF for a TCN with R residual blocks, given a vector $\mathbf{d} \in \mathbb{R}^R$ with the dilations of each block r, and a vector $\mathbf{z} \in \mathbb{R}^R$ containing the length of the filters at each block r.

$$RF = 1 + \sum_{r=0}^{R-1} \mathbf{d}[r](\mathbf{z}[r] - 1) \tag{2.15}$$

Fig. 2.3 presents a simplified representation of a TCN model. $\mathbf{X}(n, t)$ is a 2D-input with one of its axis representing time, and $\mathbf{x}_t$ is the vector of $\mathbf{X}$ at time-frame t. The network includes two residual blocks with $\mathbf{d} = [1, 2]$ and non-causal filters of length $\mathbf{z} = [3, 3]$. Following Eq. 2.15, we obtain a receptive field at the neurons of the second residual block of 7 time-frames.

2.1.2 Evaluation metrics

In segmentation and detection tasks, both the annotations of an evaluation dataset and the output of the algorithm under evaluation consist of a set of time intervals, to which we assign a class. These time intervals may or may not overlap with each other depending on the task: segmentation tasks do not allow for overlaps, while detection tasks do. All evaluation metrics are extracted from the comparison of the time intervals in the annotations of the evaluation dataset with those output by the algorithm under evaluation. From now on, we refer to the annotations of the evaluation dataset as ground truth, and the output of the algorithm under evaluation for this dataset as output.

Every evaluation metric belongs to one of two evaluation methods: the segment-based method and the event-based method. As defined by Mesaros et al. (2016), the segment-based method uses an arbitrarily short

segment of time as its minimum unit of evaluation, while the event-based method understands audio events as its minimum unit of evaluation regardless of their duration. A segment may contain more than one ground truth and output classes if a class change happens to fall between its start and end times, or if two or more time intervals overlap with it, in the case of detection tasks. On the other hand, an event must solely have a single class assigned to it.

2.1.2.1 Segment-based metrics

The value of segment-based metrics is computed from four intermediate statistics that arise from the comparison, for each segment, of its ground truth and output classes. These intermediate statistics have a different value for each class c in a taxonomy C containing L classes. The intermediate statistics are defined as follows:

- **True Positives** for class c (TP_c): number of segments containing ground truth and output class c.
- **False Positives** for class c (FP_c): number of segments containing output class c, but not ground truth class c.
- **False Negatives** for class c (FN_c): number of segments containing ground truth class c, but not output class c.
- **True Negative** for class c (TN_c): number of segments containing neither ground truth class c nor output class c.

In Fig. 2.4, we show a comparison between ground truth and output classes using a segment-based evaluation method. A segment is the time

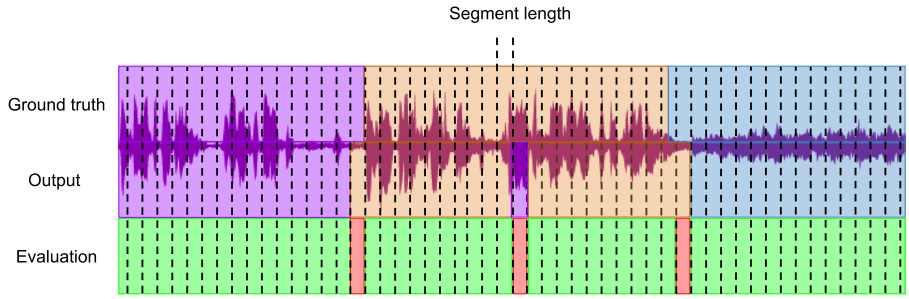


Figure 2.4: (Top) ground truth, (mid) output, and (bottom) segment-based evaluation for an audio file annotated using a taxonomy with three classes.

interval between dashed lines. In the evaluation row, every segment is either green or red, depending on the combination of its ground truth and output classes: green segments are a True Positive of the ground truth class, while red segments are a False Negative of the ground truth class and a False Positive of the output class. In addition, all segments count as a True Negative of the unrepresented classes.

Eq. 2.16 to eq. 2.18 show three very common segment-based metrics: the precision (P_c), recall (R_c), and F-measure (F_c) of each class $c \in C$. P_c is defined as the percentage of correctly classified segments of class c with respect to the total amount of segments classified as class c, and R_c is defined as the percentage of segments of class c that are correctly classified. F_c is the harmonic mean of P_c and R_c. We compute these three metrics in the same way for segmentation and detection tasks.

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{2.16}$$

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{2.17}$$

$$F_c = \frac{2P_cR_c}{P_c + R_c} \tag{2.18}$$

Accuracy (Acc) is also a very common segment-based metric that measures the percentage of correct segments with respect to the total amount of segments. Some authors opt to use the error rate (ER) instead, which is the percentage of incorrect segments with respect to the total amount of segments. Eq. 2.19 and eq. 2.20 show the formal definition of these metrics:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.19}$$

$$ER = \frac{FP + FN}{TP + TN + FP + FN} = 1 - Acc \tag{2.20}$$

where

$$\begin{aligned} TP &= \sum_{c \in C} TP_c \ , \quad & FP &= \sum_{c \in C} FP_c \\ FN &= \sum_{c \in C} FN_c \ , \quad & TN &= \sum_{c \in C} TN_c \end{aligned} \tag{2.21}$$

Note that if a class c is very sparse in the evaluation dataset and an algorithm never detects it, TN_c becomes much larger than TP_c, and contributes to a high Acc value that fails to represent the actual performance of that algorithm. Strong class imbalances also affect the P_c value of a class c with little representation in the ground truth. This happens be-

Confusion matrix			
Ground truth	**Classified as**		
	Fg	Bg	No
Fg	20.0	10.0	0.0
Bg	6.0	60.0	4.0
No	0.0	10.0	90.0

Table 2.1: Distribution by ground truth and output class of the total content of a 200 seconds mock dataset. *Fg*, *Bg*, and *No* stand for the *Foreground Music*, *Background Music*, and *No Music* classes.

cause the value of TP_c is limited by the amount of content of class c, while FP_c can take much larger values, as it depends on the amount of content of other more represented classes.

Authors such as Lu et al. (2001) and Richard et al. (2007) use what is called a confusion matrix, which cannot be described as a metric, but is nevertheless a rather useful evaluation tool that helps to differentiate between types of error and detect possible biases of an algorithm towards the classification of a certain class. If L is the number of classes in a taxonomy C, a confusion matrix must have size $L \times L$ to store the distribution of the segments according to their ground truth class (rows) and output class (columns). In Table 2.1 we show the 3×3 confusion matrix of a mock dataset with relative music loudness estimation annotations.

Weighting each row by the total number of seconds of the corresponding class yields the balanced confusion matrix $\mathbf{CM}^b$, which we depict in Table 2.2. Eq. 2.22, shows how to extract the balanced accuracy (Acc^b) from $\mathbf{CM}^b$. Eq. 2.23 and eq. 2.24 do the same for the balanced precision (P_c^b) and recall (R_c^b) of each class $c \in C$, respectively. These metrics describe the hypothetical performance of the algorithm if the evaluation dataset were balanced. In other words, it equalizes the impact on metrics

Balance-independent confusion matrix			
Ground truth	**Classified as**		
	Fg	Bg	No
Fg	66.7%	33.3%	0%
Bg	8.6%	85.7%	5.7%
No	0%	10%	90%

Table 2.2: Balance-independent distribution by ground truth and output class of the total content of a 200 second mock dataset with respect to its duration. *Fg*, *Bg*, and *No* stand for the *Foreground Music*, *Background Music*, and *No Music* classes.

of all classes, preventing algorithms that are biased towards the most common classes from obtaining the best results. Authors such as Doukhan and Carrive (2017) use this kind of metrics, and we also do in chapters 4 and 5.

$$Acc^b = \frac{trace(\mathbf{CM}^b)}{L} \tag{2.22}$$

$$P_c^b = \frac{\mathbf{CM}_{c,c}^b}{\sum_{i \in C} \mathbf{CM}_{i,c}^b} \tag{2.23}$$

$$R_c^b = \mathbf{CM}_{c,c}^b \tag{2.24}$$

Note that if a taxonomy cannot provide a class for the entirety of the audio in the dataset, it is possible to include a class such as *Other* to represent the non-annotated audio in the confusion matrix. Notice also that to use confusion matrices for detection tasks with a number of classes $L > 1$ in the taxonomy C, it is necessary to define a 2×2 confusion matrix for each class $c \in C$, using the presence of c and the absence of c

as classes. For instance, to use confusion matrices during the evaluation of a music and speech detection algorithm, it is necessary to create two 2×2 confusion matrices with classes *Music* and *No Music*, and *Speech* and *No Speech*.

2.1.2.2 Event-based metrics

Similarly to the segment-based metrics, computing the value of some event-based metrics requires the previous calculation of intermediate statistics: TP_c, FP_c and FN_c for each class $c \in C$. In the event-based evaluation method, TN_c are not as relevant, because only the presence of ground truth and output events is interesting, not their absence. The definitions of these intermediate statistics are slightly different:

- **True Positives** for class c (TP_c): the number of output events of class c that have a corresponding ground truth event of the same class.
- **False Positives** for class c (FP_c): the number of output events of class c that do not have a corresponding ground truth event of the same class.
- **False Negatives** for class c (FN_c): the number of ground truth events of class c that do not have a corresponding output event of the same class.

Mesaros et al. (2016) proposed two conditions to determine whether or not an output event has a corresponding ground truth event and vice-versa: the time interval between the onsets and/or offsets of two corresponding events must not exceed a certain tolerance window, and the du-

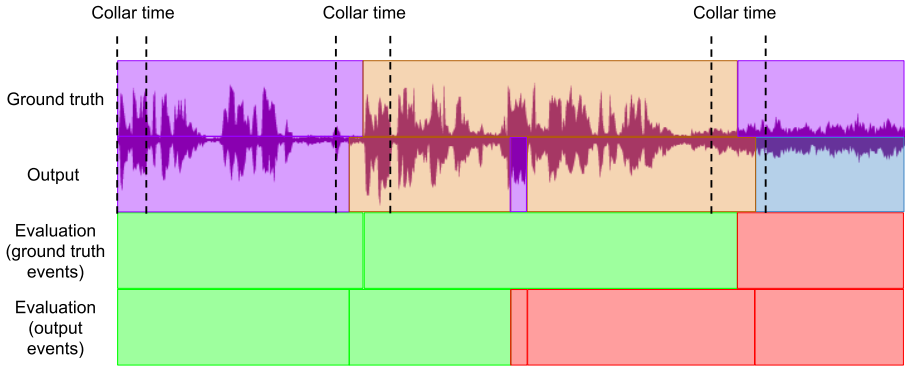


Figure 2.5: (Top) ground truth, (mid) output, and (bottom) event-based evaluation for an audio file annotated using a taxonomy with three classes.

ration of the output event must be inside certain percentages of the duration of the ground truth event. We call these conditions correspondence conditions.

In Fig. 2.5, we show a comparison between ground truth and output classes using an event-based evaluation method. An event is the time interval demarcated by each of the colored blocks in the ground truth and classification rows. Note that there are two evaluation rows: one for the ground truth events, and the other one for the output events. Green events represent a True Positive in both rows, while red events represent a False Negative in the ground truth evaluation row and a False Positive in the output evaluation row. In this example, we only impose a tolerance window for the onset of the events as the correspondence condition.

Precision (P_c), recall (R_c) and F-measure (F_c) are calculated using equations eq. 2.16 to 2.18 as their segment-based counterparts. The definitions of these event-based metrics are also the same, but using events

instead of segments.

In their work, Butko and Nadeu (2011) defined a new event-based metric called error rate (ER), which is different from the segment-based error rate metric defined above. This ER is defined as the average of the error rate of each class $c \in C$ (ER_c). ER_c represents the ratio between the errors for the class c, and the total duration of the audio annotated as class c. Eq. 2.25 and eq. 2.26 formally describe this metric:

$$ER_c = \frac{FP_c + FN_c}{TP_c + FN_c} \tag{2.25}$$

$$ER = \frac{1}{L} \sum_{c \in C} ER_c \tag{2.26}$$

Gimeno et al. (2018) propose another event-based metric called segmentation error rate (SER). To calculate this, the evaluation dataset is divided into N contiguous time intervals, and for each time interval $n \in \{0 \ldots N-1\}$ a scalar value E_n is computed, as shown in eq. 2.27, where T_n represents the duration of time interval n.

$$E_n = T_n[max(TP_n + FN_n, TP_n + FP_n) - TP_n] \tag{2.27}$$

Where

$$TP_n = \sum_{c \in C} TP_c \ , \quad for\ time\ interval\ n \tag{2.28}$$

$$FP_n = \sum_{c \in C} FP_c \ , \quad for\ time\ interval\ n \tag{2.29}$$

$$FN_n = \sum_{c \in C} FN_c \ , \qquad for\ time\ interval\ n \tag{2.30}$$

Then, adding the value of E_n for the N intervals and dividing it by the added duration of the N time intervals the value of SER is obtained.

$$SER = \frac{\sum_{n \in N} E_n}{\sum_{n \in N} T_n} \tag{2.31}$$

While segment-based metrics are very well established and their usage is broadly extended, there is no clear standard for event-based metrics, and only a few authors use them. Those who do, either use the traditional precision, recall and F-measure metrics and propose their own correspondence conditions between output and ground truth events, or directly define their own metrics as we have seen above.

A fundamental difference between segment-based and event-based metrics is that, in general terms, segment-based metrics focus solely on whether each segment has been correctly classified or not. On the other hand, current event-based metrics measure not one but two characteristics of an algorithm's output at the same time: the position of these events in time, and their class. In this way, the errors of one characteristic affect the other, creating ambiguity in the meaning of the metrics. In Fig. 2.5, we show how the presence of the smallest output event, in the middle of the second row, breaks a longer event in two parts. Only the first part is considered to be correct according to the correspondence conditions, despite the second part mostly containing the correct class.

2.2 Available resources

In the first part of this section, we detail the public and private available datasets for music detection related tasks. We also point out their shortcomings, and extract conclusions about the content and annotation requirements that future datasets should comply with. In the second part of the section, we offer an overview of the audio annotation tools that can be used to annotate this kind of datasets.

2.2.1 Datasets

Datasets are fundamental for the development of research in any task. They are needed to evaluate and compare the performance of the algorithms that we design to solves these tasks, and to keep track of their evolution over time. Moreover, in the case of machine learning algorithms, datasets are also used for training. In this section, we first describe the existing datasets that have previously been used for tasks that involve the detection of music, and then we present their shortcomings as music detection datasets.

2.2.1.1 Description

Scheirer and Slaney (1997) published the Scheirer-Slaney music/speech corpus[1] (SSMSC), the first public dataset that includes annotations about the presence of music. The taxonomy, however, is designed for the task of discriminating music and speech and includes four classes: *Music*, *Speech*, *Simultaneous Music and Speech* and *Other*. The dataset contains

[1] `https://labrosa.ee.columbia.edu/sounds/musp/scheislan.html`

245 manually annotated 15-seconds audio files digitally sampled from an FM tuner from the San Francisco Bay area. Every audio file belongs exclusively to one class, and they are separated in a training set of 184 files and a testing set of 61 files. The training set is divided into 60 files of *Music*, 60 files of *Simultaneous Music and Speech*, 60 files of *Speech* and four files of *Other*. The testing set is divided into 20 files of *Music* with vocals, 21 files of *Music* without vocals and 20 files of *Speech*.

ESTER[2] (Gravier et al., 2004; Galliano et al., 2005) is a dataset containing audio from French radio streams that was created to promote the development of speech transcription algorithms. Despite this, it also contains annotations with a taxonomy consisting of three classes: *Speech*, *Speech over Music* and *Music*. The original purpose of the dataset is very apparent in the imbalance of these three classes: from the total 100 hours of audio that form the dataset, 86% is annotated as *Speech*, 13% as *Speech over Music* and less than 1% as *Music*. This dataset is private and can be purchased for research and commercial purposes.

Seyerlehner et al. (2007) presented the first and only public dataset specific to music detection[3] (SMD) until now, i.e., with a taxonomy including only the classes *Music* and *No Music*. It has a total duration of nine hours unevenly distributed over 13 manually annotated audio files of duration between 6 and 90 minutes. The audio was extracted from TV programs of the austrian national broadcasting corporation (ORF). As shown in Table 2.3, the music content of each file ranges from 1.5% to 80%, which results in 42% of the dataset containing music.

[2] https://catalogue.elra.info/en-us/repository/browse/ELRA-S0338/

[3] http://www.seyerlehner.info/download/music_detection_dataset_dafx_07.zip

File name	Program type	Music (%)	Music (min)
Der Volksanwalt	law show	1.48	35
Starmania	music show	50.18	89
Sturm der Liebe	soap opera	70.52	49
Alpen Donau Adria	documentary	57.08	30
Barbara Karlich Show	talk show	7.51	57
Da wo es noch Treue gibt	soap opera	62.9	89
Frisch gekocht	cooking show	10.01	24
Gut beraten Österreich	talk show	8.76	18
Heilige Orte	documentary	54.34	44
Heimat fremde Heimat	documentary	29.72	30
Hohes Haus	parliament show	17.5	30
Julia	soap opera	80.36	43
ZIB	news show	4.91	7
total	-	41.99	545

Table 2.3: Music content by file in the SMD dataset.

Butko and Nadeu (2011) produced the dataset that was used for the evaluation campaign Albayzín-2010 (Albayzín-2010). The task under evaluation was the segmentation of audio in five classes: *Music*, *Speech*, *Speech over Music*, *Speech over Noise* and *Other*. The dataset includes 24 audio files of approximately four hours of duration each, which amounts to a total duration of approximately 87 hours, extracted from a Catalan news TV channel. The proportion of each class in the ground truth is: 37% of *Speech*, 5% of *Music*, 15% of *Speech over Music*, 40% of *Speech over Noise* and 3% of *Other*.

Tzanetakis generated a public dataset for the task of music and speech discrimination named GTZAN music/speech collection (GTZAN).[4] It

[4] `opihi.cs.uvic.ca/sound/music_speech.tar.gz`

consists of 128 manually annotated 30-seconds audio files. These audio files are divided equally between the classes *Music* and *Speech*. Some of the present music genres are classical music, folk music, jazz, pop, rock and electronic music. With regards to the speech part, most of the content is in English, but other languages such as German, Chinese, Greek or Serbian appear too.

MUSAN[5] (Snyder et al., 2015) is a public dataset that contains: 60 hours of speech coming from Librivox[6] and archived US government files, which are unevenly distributed between 12 different languages; 42 hours of varied music styles such as baroque, classical, romantic, country, hip-hop, jazz, etc.; and six hours of technical and non-technical noises including, for instance, sounds of nature and the city. Every file belongs exclusively to one of the following classes: *Music*, *Speech* and *Noise*. The dataset includes annotations of the musical genre in the case of the music files and annotations of the speaker language and sex in the case of the speech files.

The MuSpeak Team[7] at City University London annotated a dataset (MIREX15) that was proposed as training data for the music information retrieval evaluation exchange (MIREX) tasks of music/speech classification and detection of 2015.[8] The dataset consists of seven audio files with durations between 28 and 65 minutes containing concerts and musical radio programs featuring four musical genres: classical music, country music, ethnic music, and Irish music. The dataset is annotated using the classes *Music* and *Speech*. The annotations contain 75.7% of isolated *Mu-*

[5] http://www.openslr.org/resources/17/musan.tar.gz
[6] librivox.org
[7] http://mirg.city.ac.uk/muspeak
[8] http://mirg.city.ac.uk/datasets/muspeak/muspeak-mirex2015-detection-examples.zip

sic, 22.1% of isolated *Speech*, and 2.2% of overlapped *Music* and *Speech*.

2.2.1.2 Analysis

We perceive many shortcomings in the datasets described in the previous section. We consider that datasets suitable for music detection related tasks in the context of broadcast audio must fulfill three fundamental requirements: (1) a significant amount of audio files must contain class changes so that the precision of an algorithm in detecting them can be measured; (2) the dataset must contain both isolated music and music mixed with other type of sounds; and (3) these non-music sounds must include but not be limited to speech. On top of that, it is highly desirable for broadcast audio datasets to contain a rich variety of scenarios, including different program types and languages, so that they are as representative of reality as possible. Needless to say, the more audio contained in the dataset the better.

Of all the aforementioned datasets, only the SMD dataset fulfills these three requirements. Unfortunately, it only contains nine hours of audio, and the audio comes from a single broadcaster, which severely limits how representative it is of the diversity of broadcast audio. In the SSMSC, ESTER, and Albayzín-2010 datasets, isolated and mixed music are annotated differently, however, the mixed music is limited to background music with speech as the non-music part. Moreover, in the case of SSMSC, the audio files are very short and belong entirely to a single class. The GTZAN and MUSAN datasets do not contain mixed music and include only audio files of a single class. We show a comparison between the characteristics of these datasets in Table 2.4.

Dataset	Duration	Number of files	Class changes	Mixed music	Relative loudness annotations
SSMSC	1 hour	245	No	With speech	Speech over music
SMD	9 hours	13	Yes	Yes	No
GTZAN	1 hour	128	No	No	No
MUSAN	108 hours	2016	No	No	No
Albayzín-2010	87 hours	24	Yes	With speech	Speech over music
ESTER	100 hours	144	Yes	With speech	Speech over music
MIREX15	5 hours	7	Yes	With speech	Speech over music

Table 2.4: Comparison of the characteristics of the datasets for music detection related tasks.

We conclude that there is a need for new public datasets for music detection related tasks that are designed to solve the aforementioned shortcomings: these datasets must be varied in terms of broadcast scenarios, their audio files must include class changes, they must contain both isolated and mixed music, and the mixed music must be combined with all kinds of non-music sounds. If we want to make these datasets suitable to the task of relative music loudness estimation, we also need information about the loudness of music relative to these non-music sounds. Note that music detection algorithms can also make use of this relative loudness information, especially for error analysis. This conclusion has led us to the creation of the OpenBMAT dataset, which we present in Section 4.2.

2.2.2 Audio annotation tools

To generate datasets, we need both data and a way to annotate it. When working with audio, this is typically done using audio annotation tools. In this section, we provide an overview of the annotation tools that can be used to generate annotations about the presence of music.

Sjölander and Beskow (2000) built one of the first tools for the annotation of audio named WaveSurfer[9]. WaveSurfer was originally created to produce annotations related to speech, but was deliberately designed to be flexible and extensible to other tasks. And indeed, it was used after that, for instance, by Herrera et al. (2005) to develop MUCOSA, an environment for the annotation and generation of music metadata at different abstraction levels that incorporated a collaborative annotation subsystem. Praat[10] (Boersma, 2001) was also originally meant for the annotation of

[9] http://www.speech.kth.se/wavesurfer/
[10] http://www.fon.hum.uva.nl/praat/

speech, but has been used to annotate musical content as well.

With a wider range of applications in mind, Cannam et al. (2006) designed Sonic Visualizer, which is well-known for its varied analysis and feature extraction functionalities, which are added to the tool using Vamp plug-ins. The same year, Wittenburg et al. (2006) built ELAN, a full-featured and complex tool that allows the annotation of both audio and video. Krijnders and Andringa (2009) created SoundScape, a tool that introduces the use of machine learning algorithms to reduce the annotation time. Specifically, it incorporates an algorithm that suggests possible annotations and presents them to the annotators for acceptance. Any accepted or corrected annotation is used to further improve the classifier. This tool also allows the annotation of specific time-frequency regions of the spectrogram.

Cartwright et al. (2017) designed audio-annotator,[11] which is an open-source JavaScript web front-end interface for the annotation of audio events that uses and extends wavesurfer.js[12]. Wavesurfer.js allows for the playback of audio, the visualization of its waveform, and the selection of time intervals on top of it. The extended version includes features such as labeled time intervals, or the possibility to switch the audio visualization between its waveform and its spectrogram.

Despite the variety of available audio annotation tools, none of them meet the requirements that we later specify in Section 3.1. This leads us to the creation of our own tool. Nevertheless, audio-annotator includes several features that we consider useful for the annotation of audio events. In Section 3.2, we explain how we incorporate them into our own tool.

[11] `https://github.com/CrowdCurio/audio-annotator`
[12] `https://wavesurfer-js.org/`

2.3 Tasks and methods

This section contains the literature review about music detection related tasks. We divide it into two parts: first, we describe the algorithms that deal with the task of music detection as a binary-class task; and then, we review the algorithms that combine the detection of music with the detection of speech, and occasionally also other specific types of non-music sounds. In each section, we divide algorithms into two groups: those that rely on feature engineering, and those that learn features automatically from a representation of the audio using deep learning.

2.3.1 Music detection as a binary-class task

Music detection consists of the detection of time intervals that contain music in an audio file, or, in other words, the segmentation of an audio file in time intervals of the *Music* and *No Music* classes. In this task, foreground and background music are not considered two different classes; nevertheless, many authors differentiate between foreground and background music in their works: Seyerlehner et al. (2007) mentioned these two concepts while stating that background music is harder to detect. Several other authors, including Zhu et al. (2006), Izumitani et al. (2008), and Giannakopoulos et al. (2008), agree that music detection is often applied to scenarios characterized by a strong presence of background music, such as TV broadcast audio, effectively differentiating them from scenarios where most of the time music has the main role.

Category	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$
News	94%	96.5%
Commercial	99%	91%
Drama	98.5%	98%
Total	96%	95.5%

Table 2.5: Results of the algorithm proposed by Zhu et al. (2006) for a testing dataset based on TRECVID 2005.

2.3.1.1 Feature engineering approaches

To the best of our knowledge, Zhu et al. (2006) proposed the first method that is specific to music detection. The method is based on the analysis of the temporal evolution of the audio's spectrum in order to detect the presence of a tuning frequency and recognize its temporal continuity. According to the authors, the presence of music concentrates the energy of the spectrum in very specific frequencies, which promotes the appearance of a clear tuning frequency, while other sounds produce a more even distribution. In this way, a constant tuning frequency indicates the presence of music, and the degree of energy concentration helps to distinguish between isolated music and music mixed with other sounds.

Table 2.5 shows the segment-based precision and recall values obtained by the algorithm using four hours of audio from the Chinese TV channel CCTV as testing dataset. The audio was extracted from the TRECVID 2005 dataset and contains mainly news and drama programs, as well as commercials. The authors manually produced annotations about the presence of music for this content as the TRECVID 2005 dataset does not include them. Unfortunately, they did not specify the exact audios they used nor did they provide access to the annotations. In the conclusions, the paper points out that this method is not valid for percussion

music as percussion instruments do not have a clear pitch.

One year later, Seyerlehner et al. (2007) designed a feature called continuous frequency activation (CFA) following the idea that the presence of music creates certain recognizable patterns in the spectrogram. The CFA is computed from the frequency activation function, which is a function that represents the time each frequency is active during a particular time interval. The calculation of the CFA consists of the addition of the five highest height-to-width ratios of all the peaks of this function.

Figure 2.6 shows two examples of a frequency activation function. At the top part of the figure, the authors show, from left to right, the spectrogram and the frequency activation function of an audio that does not contain music. At the bottom, we see the same plots but for an audio containing music. In summary, the CFA detects strong activations of isolated frequencies that are sustained in time. The authors claimed that this is usually a phenomenon generated by the presence of music, and thus, the CFA is a feature suitable for the task of music detection. Using this feature and a simple threshold, the authors reached 89.9% accuracy for the SMD.

Even though the previous authors implicitly took advantage of temporal context information through the study of the evolution in time of the spectrum’s energy, Giannakopoulos et al. (2008) were the first authors to explicitly mention that there is a certain dependency among successive time-frames of broadcast audio that can be exploited to enhance the performance of algorithms for the task of music detection. To do that, their method summarizes the evolution of a set of four features across short-term windows into a mid-term value. Two of these features are chroma-based, i.e., they are related to the chromagram, and the other two features

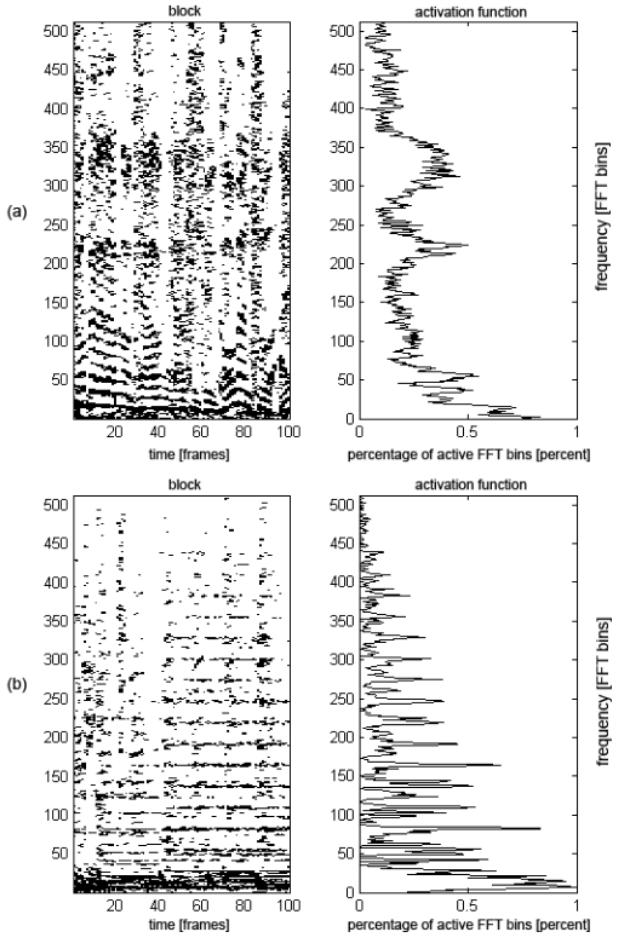


Figure 2.6: Two frequency activation functions: (a) one for an audio excerpt that does not contain music and (b) the other for an audio excerpt that contains music. Reprinted from Seyerlehner, K., Pohle, T., Schedl, M., and Widmer, G. (2007). Automatic music detection in television productions. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07).*

are related to the energy entropy and the presence of a recognizable pitch. The chromagram, as originally defined by Bartsch and Wakefield (2005), is a time-frequency representation of audio that shows the distribution of the spectrum's energy among each of the 12 pitch classes of the twelve-tone equal-tempered scale at each time-frame. The first chroma-based feature is designed to exhibit higher values when there are clearly dominant pitch classes, which, according to the authors, is an indicator of the presence of music. The second chroma-based feature measures the variation of each pitch class over time. The authors claimed that when music is present there is always at least one pitch class that is very stable, while in the case of speech all elements show high variability.

The algorithm uses the training set to generate two histograms for each feature: one for the *Music* class, and a second one for the *No Music* class. During testing, the algorithm extracts these features for a particular time-frame and, after comparing them with the corresponding histograms, it outputs the most probable class for that time-frame. The audio in both training and testing datasets comes from several movies and it was cross-annotated by three annotators. The authors considered that all annotators must agree on the presence of music for a *Music* time interval to appear in the final ground truth. In our opinion, this can easily lead to the omission of time intervals of background music with very low volume, artificially improving the algorithm's performance. We suppose that the authors could not provide access to the data due to copyright reasons.

In Table 2.6 presents the performance of the algorithm using both the segment- and event-based evaluation methods. Note that the authors used a rather lax correspondence condition: an output event needs only to overlap with a ground truth event of the same class to be considered correct.

Evaluation method	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$
Segment-based	89%	83%	86%
Event-based	91%	90%	90.5%

Table 2.6: Results of the algorithm proposed by Giannakopoulos et al. (2008) for a testing dataset that consists of excerpts of several movies.

In their work, Izumitani et al. (2008) explicitly stated that background music is both a strong challenge for music detection algorithms, and a common type of content in broadcast audio. Focusing on its detection, they proposed a comparison between different sets of features to find out which set is the most appropriate. They trained k-nearest neighbors (kNN) and gaussian mixture models (GMM) algorithms using four sets of features. The first set includes what the authors called the spectral powers from a linear-scaled spectrogram. To compute these, the spectrogram is first divided into a series of frequency bands, and then the average of the values that fall inside each band is computed. The second set consists of the Mel-scaled version of the first set. The third and fourth sets include, respectively, the Mel-frequency cepstrum coefficients (MFCC), and seven engineered features that include the zero-crossing rate, the 4 Hz modulation energy, which is higher for audio containing speech (Houtgast and Steeneken, 1973), the percentage of low-energy frames, and four low-level spectral features.

The authors used a synthetic testing dataset containing mixes of music and speech at different signal-to-noise ratio (SNR) with the music as the signal and the speech as the noise. The music was extracted from the RWC Music Database,[13] and the speech from the Corpus of Sponta-

[13] `https://staff.aist.go.jp/m.goto/RWC-MDB/`

neous Japanese.[14] Both datasets can be obtained free of charge through an access request or a registration; however, the authors do not provide information about the specific files that they mixed. The authors reported an error rate of 8% for the best combination of algorithm and feature set, and an SNR of -10 dB. This percentage increases to over 20% for an SNR value of -20 dB even though at this SNR level music is still easy to hear. We want to highlight that this is the only work presenting a music detection algorithm that provides information about its performance for the specific case of background music.

2.3.1.2 Deep learning approaches

Gfeller et al. (2017) described an audio fingerprinting application for mobile devices that uses music detection to reduce the amount of audio to analyze. This is an example of the application of music detection as a preprocessing step to improve the performance of an algorithm designed for another task. The authors presented a CNN including six consecutive 2D-convolutional layers with max-pooling and two dense layers. They trained this network using log-magnitude Mel-scale features extracted from a subset of AudioSet (Gemmeke et al., 2017), and they evaluated it against 450 hours of audio of unknown origin divided into about 12 thousand time intervals of durations between 16 and 40 seconds containing music with a loudness ranging from imperceivable by humans to very loud. They achieved a music segment-based recall of 75.5% when accepting a false positive every 20 minutes in average on non-silent audio.

Jang et al. (2019) proposed a new type of 2D-convolutional filter for music detection called *melCL* filter. As shown in Figure 2.7, it mimics

[14] `https://pj.ninjal.ac.jp/corpus_center/csj/en/`

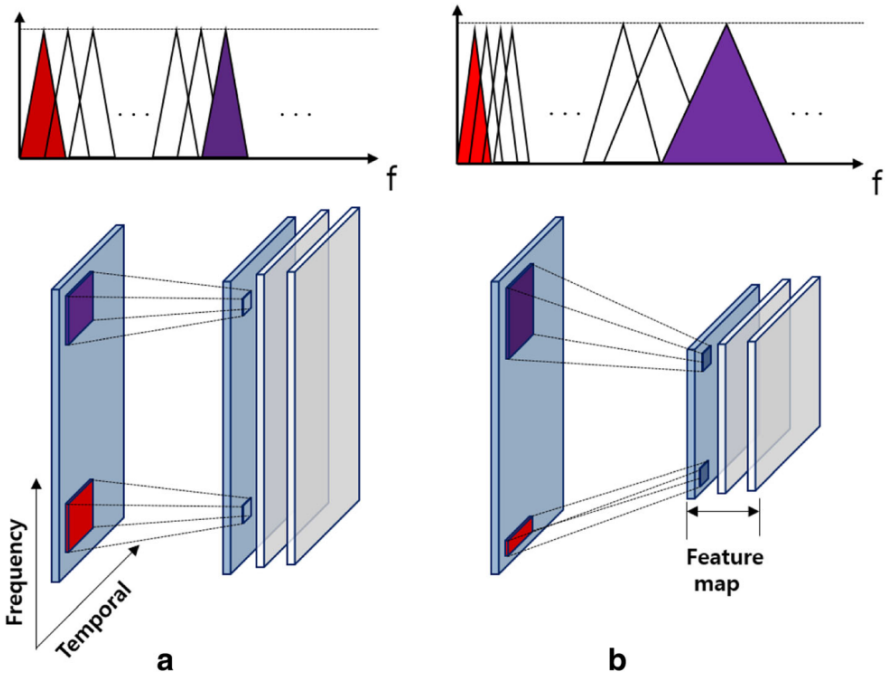


Figure 2.7: Convolutional layers with (a) a fixed-size kernel and (b) a Mel-scale kernel. Reprinted from Jang, B.-Y., Heo, W.-H., Kim, J.-H., and Kwon, O.-W. (2019). Music detection from broadcast contents using convolutional neural networks with a Mel-scale kernel. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):11.

the filters used in a Mel filter bank with the advantage of having trainable weights. The authors presented a network that concatenates one Mel-scale 2D-convolution layer, which is a 2D-convolutional layer that uses melCL filters, three regular 2D-convolutional layers with max-pooling, and two dense layers. We show this network in Fig. 2.8. It takes the log-magnitude power spectrogram as input, and outputs the probability of *Music* and *No Music* at a frame level through an output layer with two neurons and a

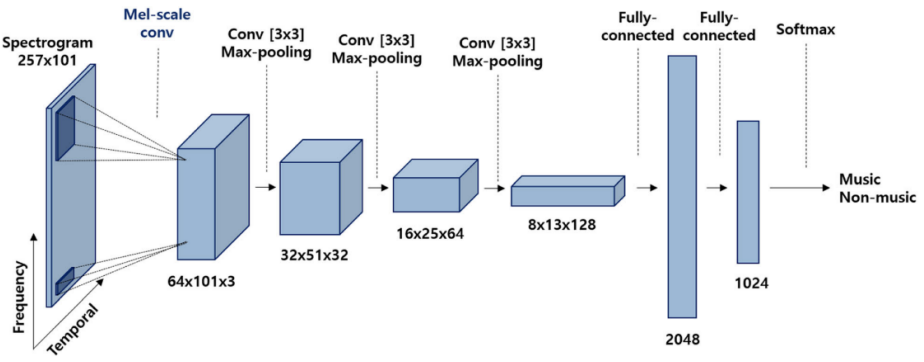


Figure 2.8: CNN architecture proposed by Jang et al. (2019). Reprinted from Jang, B.-Y., Heo, W.-H., Kim, J.-H., and Kwon, O.-W. (2019). Music detection from broadcast contents using convolutional neural networks with a Mel-scale kernel. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):11.

softmax activation function. The softmax function produces a probability distribution $\sigma_{softmax}(\mathbf{v}) \in \mathbb{R}^L$ over the L classes of a taxonomy, from a vector $\mathbf{v} \in \mathbb{R}^L$, as we show in Eq. 2.32.

$$\sigma_{softmax}(\mathbf{v}) = \frac{e^{\mathbf{v}}}{\sum_{l=1}^{L} e^{\mathbf{v}_l}} \tag{2.32}$$

The paper compares the proposed model with two published algorithms (Doukhan and Carrive, 2017; Tsipas et al., 2017), and with other CNNs, as well as with two types of RNN (Rumelhart et al., 1986). RNNs are a type of network characterized by their ability to model temporal context, which is a desirable characteristic when working with temporally correlated signals such as music. Broadly speaking, this is achieved by feeding the input to the network in successive time steps. At each time step, every neuron incorporates its past outputs into its current input. As

shown in eq. 2.33, now the output of layer l at the current time step $\mathbf{h}_t^{(l)}$ is the result of a function ϕ that depends on the output of the previous layer at the current time step $\mathbf{h}_t^{(l-1)}$, the output of the current layer in the last time step $\mathbf{h}_{t-1}^{(l)}$, and the weights and biases of the current layer $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$. σ represents the activation function.

$$\mathbf{h}_t^{(l)} = \sigma(\phi(\mathbf{h}_t^{(l-1)}, \mathbf{h}_{t-1}^{(l)}, \mathbf{W}^{(l)}, \mathbf{b}^{(l)})) \tag{2.33}$$

The authors specifically used bidirectional long short-term memory (BiLSTM) networks and bidirectional gate recurrent unit (BiGRU) networks. long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 1999) and gate recurrent unit (GRU) (Cho et al., 2014) networks are types of RNN containing gated memory cells that are able to learn and retain information in long dependencies without suffering from the vanishing/exploding gradient issue that affects simpler RNNs (Bengio et al., 1994). The bidirectional variant of these networks allows the training sequence to be read forward and backward including both past and future information into the network decisions.

The training dataset was synthetically created by mixing music, speech and noise. To do this, the authors used 25 hours of library music, 25 hours of speech from the MUSAN dataset, and almost three hours of noise from the ESC-50 database (Piczak, 2015). For testing, they used several datasets: eight hours of broadcast audio in British English (British), and 12 hours of broadcast audio in Spanish (Spanish), both including different types of program; as well as three hours of Korean drama, which was used as the validation set during training (Korean dev), and 1.5 hours of reality shows also in Korean (Korean). In the paper, they highlighted the presence of background music and sound effects in

Dataset	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$	$\mathbf{R_{mu\_nosp}}$
Korean dev	95.9%	96%	95.9%	96.8%
Korean	93%	96.4%	94.7%	97.5%
British	85.3%	87.8%	86.5%	92.2%
Spanish	84.7%	93.4%	88.9%	96.8%
MIREX15	99.4%	91.6%	95.3%	92.5%
SMD	88.3%	87%	87.6%	87%

Table 2.7: Results of the algorithm proposed by Jang et al. (2019) for six datasets.

these datasets. They also included the MIREX15 and SMD datasets as evaluation datasets. With regards to the MIREX15 dataset, the authors emphasized that it barely contained any background music.

The authors annotated all the audio for testing using Praat and a taxonomy containing the classes *Music* and *Speech*. They included the *Speech* class to also report the recall of the *Music* class in the absence of speech (R_{mu_nosp}). The proposed model obtains the best results for all testing datasets except for MIREX15, for which both Tsipas et al. (2017) and Doukhan and Carrive (2017) achieved slightly better segment-based *Music* F-measure. Table 2.7 shows the results of the proposed algorithm for all the testing datasets. We observe that $\mathbf{R_{mu\_nosp}} > \mathbf{R_{music}}$ for all datasets except for the SMD dataset where both metrics have the same value. This shows that it is harder to detect music when it is mixed with speech, and possibly other non-music sounds as well.

2.3.2 Music detection in multi-class tasks

Besides the binary-class approach to music detection, the literature also addresses the detection of music combined with the detection of speech,

and sometimes other types of sounds such as noise or environmental sounds, or even silence. In the early stages of the research in this field, the main task was music and speech segmentation, which consists of dividing an audio recording into non-overlapping time intervals of these two classes. As mentioned in Section 2.1.2, segmentation tasks do not allow for time intervals of different classes to overlap. It can easily be seen, therefore, that this task is barely applicable to the context of broadcast audio, as it assumes that music and speech cannot overlap, which is radically false in this context. This task would later evolve in two different ways to solve this issue: the first way stuck to the segmentation approach while introducing a new class to represent overlaps between music and speech, and the second way redefined the task from a segmentation approach to a detection approach, which does allow for overlaps between time intervals of different class. In this section, we do not include algorithms that are restricted to the assignment of a single class to an entire audio file. We consider this to be an oversimplification of the tasks that this thesis is concerned with, and a task that is of limited use in the context of broadcast audio monitoring for copyright management applications. On top of that, it can be approached as a summary of the results of the segmentation and detection tasks.

2.3.2.1 Feature engineering approaches

Saunders (1996) became the first author to publish a paper describing a music and speech segmentation algorithm. This algorithm is based on a multivariate-Gaussian classifier that relies on the zero-crossing rate and features related to the energy contour of the audio waveform. To test his algorithm, the author used radio audio about which he did not give further

information. He claimed to achieve 98% accuracy using this data.

Scheirer and Slaney (1997) extended the pool of features up to eight including the energy modulation at 4 Hz, and low-level features such as the spectral flux or the spectral centroid, among others. They trained four different classifiers with these features, the best of which is a spatial partitioning scheme based on k-d trees that obtains an average segment-based error rate of 5.7%. The authors used the training and testing splits of the SSMSC dataset.

Taking the possibility of overlap between music and speech into consideration, Scheirer and Slaney (1997) briefly explained, in the conclusions, that they also used these features and a simple three-way classifier to discriminate between *Speech*, *Music*, and *Simultaneous Speech and Music*. Adding this third class increases the average error rate to 35%. This last result reveals the complexity of distinguishing between these three classes even when the non-music part includes only speech and not other types of non-music sounds.

Lu et al. (2001) proposed two new classes to build a taxonomy that allows the annotation of the entire audio recording: *Environment Sound* and *Silence*. Basically, any sound that is neither music nor speech can be annotated as *Environment Sound*. However, they still failed to consider overlaps between classes. The entire classification system combines a kNN algorithm with a silence detector and a classifier based on a set of heuristic rules. First, the kNN discriminates between speech and non-speech content using information about the spectral flux, the temporal evolution of the zero-crossing rate, and the energy and envelope of the signal. Then, a silence detector relies again on the zero-crossing rate and the energy of the signal to mark the silent parts of the non-speech content.

$\mathbf{Acc}$	$\mathbf{R_{music}}$	$\mathbf{R_{speech}}$	$\mathbf{R_{env\_sounds}}$
96.5%	93%	97.5%	84.4%

Table 2.8: Results of the algorithm proposed by Lu et al. (2001) for their testing dataset.

Finally, the remaining content is divided into *Music* or *Environment Sound* by a set of heuristic rules based on the periodicity of several frequency bands, the spectral flux and the presence of noise. The audio in the testing dataset comes from the MPEG-7 dataset, and some news and movie clips as well as some audio clips from the Internet. Table 2.8 presents the performance reported by the authors.

Panagiotakis and Tziritas (2005) also relied on a set of music and speech segmentation rules that make use of the zero-crossing rate, the root mean square energy of the audio and combinations of both. These rules together with a silence detector allowed the authors to achieve segment-based recalls for the *Music* and *Speech* classes of 92% and 97%, respectively, in a testing dataset composed of audio files from the internet and recordings from various archival CDs. The authors do not provide access to this data.

Richard et al. (2007) finally set out, from the beginning, a segmentation task that uses a taxonomy including three mutually exclusive classes: *Music*, *Speech*, and *Mixed*. The authors used feature selection strategies to filter an initial set of 500 features including temporal, spectral, cepstral, and perceptual features. The algorithm they presented works as follows: first, an SVM trained with the selected features outputs a class for each time-frame; then, time-frames are grouped in time intervals by an algorithm based on novelty detection, and the most represented class inside a time interval becomes its class. The authors carry out an event-based

Balance-independent confusion matrix			
Ground Truth	**Classified As**		
	Music	Mixed	Speech
Music	47.7%	36.1%	16.1%
Mixed	4.8%	70.2%	25.0%
Speech	0.1%	2.3%	97.6%

Table 2.9: Balance-independent confusion matrix of the algorithm proposed by Richard et al. (2007) for their testing dataset.

evaluation using a tolerance window of 0.25 seconds as their correspondence condition. The dataset chosen for training and testing is the ESTER dataset. This dataset has a strong lack of music, this is why the authors chose to complement it with 40 minutes of music from the RWC dataset.

They report an overall F-measure of 96.5%, while the F-measure by class is 98.9% for the *Speech* class and 79.3% for the *Music* class. The difference in the impact on the overall F-measure of the two classes shows that this dataset is highly unbalanced towards the *Speech* class. This situation makes the overall F-measure a deceptive metric that can lead to the wrong conclusions regarding the performance of the algorithm. The authors presented the balanced confusion matrix that we show in Table 2.9. Despite the high overall F-measure, this confusion matrix reveals that their algorithm exhibits a strong bias towards the detection of *Speech*, with 16.1% of the *Music* content and 25% of the *Mixed* content classified as *Speech*, and that it has difficulties in differentiating between *Music* and *Mixed*, with 36.1% of the *Music* content classified as *Mixed*.

2.3.2.2 Deep learning approaches

Schlüter and Sonnleitner (2012) were the first authors to approach the

Dataset	**Acc**	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$
Swiss	97.3%	98.8%	98%	98.4%
Austrian	95.6%	97.3%	97.4%	97.3%

Table 2.10: Results of the algorithm proposed by Schlüter and Sonnleitner (2012) for their two testing datasets.

task of music and speech detection instead of its segmentation counterpart. The authors designed a first layer for their network that learns to extract features from the log-magnitude Mel-spectrogram in an unsupervised way. This type of layer is based on restricted Boltzmann machines (RBM) (Rumelhart and McClelland, 1987; Hinton, 2002), and it is called mean-covariance restricted Boltzmann machines (mcRBM).

To check the value of these features, the authors trained an MLP with them, as well as directly with the log-magnitude Mel-spectrogram, and also with the MFCCs. The comparison between the resulting models shows that the mcRBM features are the best option when used for speech detection. For the music detection task, though, they produce no significant performance difference with respect to the log-magnitude Mel-spectrogram. To complete the network, they attached a deep belief network (DBN), which is a stack of RBM layers, to the mcRBM layer, and added an output layer containing a single neuron with the sigmoid activation function (Han and Moraga, 1995). Eq. 2.34 shows the mathematical expression of the sigmoid activation function $\sigma_{sigmoid}$ given and input scalar $x \in \mathbb{R}$.

The authors trained one instance of this network for music detection, and the other for speech detection. They compared the music detection network against the approaches of Seyerlehner et al. (2007) and Liu et al. (2007) with a segment-based evaluation method. Their network outper-

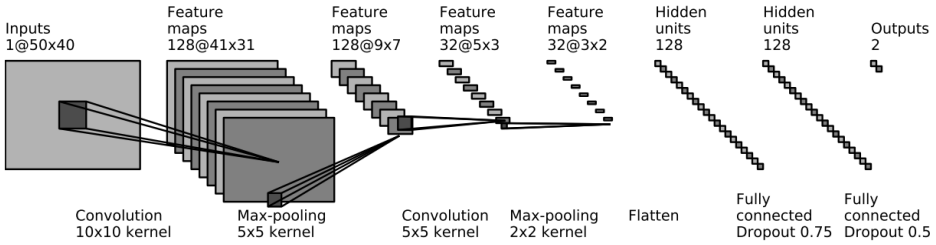


Figure 2.9: CNN architecture proposed by Doukhan and Carrive (2017). Reprinted from Doukhan, D. and Carrive, J. (2017). Investigating the use of semi-supervised convolutional neural network models for speech/music classification and segmentation. In *The Ninth International Conferences on Advances in Multimedia (MMEDIA).*

formed the other algorithms with the results shown in Table 2.10. They used two evaluation datasets: one containing 30 hours of audio from six Swiss radio web streams (Swiss), and the other with 12 hours of four Austrian web radio streams (Austrian). 15 hours of the Swiss dataset were used for training and 6 for validation, leaving 9 hours for testing.

$$\sigma_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.34}$$

Doukhan and Carrive (2017) trained several CNN architectures for the task of music and speech segmentation with different combinations of the number of 2D-convolutional and dense layers, filter shapes, pooling strategies, and regularization methods. As shown in Fig. 2.9, the CNN performing the best includes two 2D-convolutional layers, each followed by a max-pooling layer, and two dense layers with high dropout rates. The input features comprise 40 MFCCs of 50 consecutive time-frames, which is equivalent to 500 ms. The output layer has two neurons and a softmax activation function that yields the probability of each class.

Musical genre	$\mathbf{R_{music}}$	$\mathbf{R_{speech}}$
Classical	100%	100%
Country	88.55%	96.82%
Ethnic	79.26%	99.18%
Irish	96.5%	99.38%

Table 2.11: Results of the algorithm proposed by Doukhan and Carrive (2017) for the MIREX15 dataset divided by musical genre.

The authors trained the first 2D-convolutional layer apart from the rest of the network using an unsupervised strategy (Coates and Ng, 2012), to associate this layer's filters to visually relevant features such as vertical, horizontal and diagonal patterns. The weights of these filters remain constant during the training of the rest of the network. The training data consists of the combination of three public datasets: the GTZAN, the SSMSC, and the MUSAN datasets. The authors evaluated their best CNN model using the MIREX15 dataset. We show the results of this evaluation separated by musical genre in Table 2.11.

Gimeno et al. (2018) proposed the first approach to the task of music and speech segmentation that uses RNNs. They presented a model consisting of two stacked BiLSTM layers and an independent linear perceptron that transforms the output of the last BiLSTM layer into a frame-level classification. This classification is then smoothed using a hidden markov model (HMM) algorithm. The features that the authors chose for the network are a combination of the log-magnitude Mel-spectrogram, and chroma features. They train their RNN with the training split of the Albayzín-2010 dataset, and evaluate their network using two event-based metrics: the segmentation error rate (SER) and the error rate (ER), which we described in Section 2.1.2.2. The authors obtain a SER of 12.5% and

Evaluation method	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$
Segment-based	96.9%	97.3%	97.1%
Event-based (on)	77.8%	78.7%	78.2%
Event-based (on-off)	58.7%	59.3%	59%

Table 2.12: Results of the algorithm proposed by Lemaire and Holzapfel (2019) for their testing data.

an ER of 20% for the testing split of the Albaycín-2010 dataset. These results surpass those of the best algorithms evaluated so far using this dataset.

Lemaire and Holzapfel (2019) proposed, for the first time, to use TCNs for the task of music and speech detection. They published a comparison between a TCN with non-causal filters (ncTCN), and three other architectures: a concatenation of a CNN, a LSTM network, and an MLP (CLDNN); a BiLSTM network (BiLSTM); and a TCN with causal filters (TCN). They trained several models of each architecture in a grid search over a set of hyper-parameters. All architectures use the log-magnitude Mel-spectrogram as input features.

The data used for training and testing comprises several public and private datasets. The public datasets include the MUSAN, GTZAN, SSMSC, MIREX15, and ESC-50 datasets. All these data together amounts to a total duration of approximately 221 hours. The authors compared the best model of each architecture through the computation of the area under the ROC curve, and the ncTCN obtained an area value slightly above the rest. Table 2.12 shows the results for music detection of the ncTCN using both a segment-based and a event-based evaluation method, respectively. The authors used a post-processing strategy to improve the results of the network by smoothing its output based on a minimum du-

ration of the time intervals of each class, and the minimum duration of the breaks between time intervals of the same class. For the event-based evaluation, the correspondence condition is a tolerance window of 500 ms, which they apply to only the onsets (on), and both onsets and offsets (on-off).

2.3.3 Analysis

We observe that broadcast media is the main focus in terms of context of most of the previously reviewed papers. We also observe a general consensus among authors that background music is one of the most common and challenging types of content in broadcast audio: music constantly appears overlapped with speech and/or other non-music sounds, which makes it harder to detect. Despite all of this, only a few authors evaluate their algorithms specifically for this type of content (Scheirer and Slaney, 1997; Richard et al., 2007; Izumitani et al., 2008; Gimeno et al., 2018). Those who do, do not obtain outstanding results even though they only consider the particular case of speech over music, disregarding the possibility of music overlapping with any other type of non-music sound.

A serious issue that we notice across the reviewed literature is that not many algorithms are evaluated using public data. Many authors use private audio to generate or complement their testing datasets, or do not describe the origin of the audio with enough precision as to be able to reproduce these datasets. Without access to the contents of the testing datasets we cannot validate the results obtained by an algorithm, we cannot assess the true value of these results, as we have no information on how challenging the datasets are, and we cannot compare these results with other algorithms and track the evolution of the tasks. In the case

of the tasks reviewed in this thesis, the outcome of this situation is the sensation that every author has solved them on their own. However, this is far from the truth as it becomes apparent when evaluating with challenging datasets such as in MIREX 2018[15], or in papers such as that of Jang et al. (2019). Of course, having a good set of public datasets helps to promote their usage for evaluation and, as explained in Section 2.2.1.1, we perceive that this is not the case.

Another observation concerns some of the combinations taxonomy-task that appear in the reviewed literature. Many authors (Saunders, 1996; Lu et al., 2001; Panagiotakis and Tziritas, 2005; Doukhan and Carrive, 2017) design their algorithms for the task of music and speech segmentation, without taking into account the possibility that these type of sounds may overlap. In the case of Doukhan and Carrive (2017), we believe that this happens because they defined an ad hoc taxonomy for the evaluation dataset that they used, which does not contain overlaps between music and speech. We can not confirm that the same applies to the other authors because their evaluation data is private. A similar situation occurs in the work of Gimeno et al. (2018), where they use Albayzín-2010 as their evaluation dataset. Their taxonomy includes the *Speech over Music* class, and also the *Speech over Noise* class; however, it does not include, for instance, the *Music over Noise* class, which is a common combination of sounds in broadcast audio. This happens because the Albayzín-2010 dataset does not contain this type of combination. Table 2.13 shows all the combinations of taxonomies and types of task used in the approaches reviewed in Section 2.3.2. It also presents the information

[15] https://www.music-ir.org/mirex/wiki/2018:Music_and_or_Speech_Detection_Results

Authors / Task	Taxonomy	Task type	Music overlap annotations with
MD task	Music (and No Music)	Detection (Segmentation)	Generic non-music sounds
Saunders (1996)	Music and Speech	Segmentation	Nothing
Scheirer and Slaney (1997)	Music, Speech (and Speech over Music)	Segmentation	Speech
Lu et al. (2001)	Music, Speech, Environment Sound, and Silence	Segmentation	Nothing
Panagiotakis and Tziritas (2005)	Music, Speech, and Silence	Segmentation	Nothing
Richard et al. (2007)	Music, Speech, and Mixed	Segmentation	Speech
Schlüter and Sonnleitner (2012)	Music and Speech	Detection	Speech
Doukhan and Carrive (2017)	Music and Speech	Segmentation	Nothing
Gimeno et al. (2018)	Music, Speech, Speech over Music, Speech over Noise, and Other	Segmentation	Speech
Lemaire and Holzapfel (2019)	Music and Speech	Detection	Speech
RMLE task	Foreground Music, Background Music, and No Music	Segmentation	Generic non-music sounds

Table 2.13: Combinations of taxonomy and type of task used in each of the approaches described in Section 2.3.2, and the information that they include about overlaps of music and non-music sounds.

that these approaches provide about the overlaps between music and non-music sounds.

Finally, we want to underline the gradual appearance of deep learning in the reviewed literature that started with the work of Schlüter and Sonnleitner (2012). According to their work and that of authors such as Doukhan and Carrive (2017) and Gimeno et al. (2018), deep learning has overcome feature engineering approaches and also other machine learning techniques. Jang et al. (2019) compared different network architectures, showing that RNNs such as LSTM and GRU networks, and their bidirectional counterparts, generally offer worse results, in comparison to CNNs, despite their capacity to model temporal context, which should be a valuable characteristic when dealing with temporally correlated signals such as music. Lemaire and Holzapfel (2019) proposed the usage of TCNs, which is a type of CNN that can also model temporal context, does not suffer from vanishing/exploding gradients as some RNNs do (Bengio et al., 1994), and have proven to produce better results than RNNs (Bai et al., 2018). Finally, Lemaire and Holzapfel (2019) and other authors such as de Benito-Gorron et al. (2019) showed that concatenating a CNN with an RNN can boost the performance of the RNN and offer state-of-the-art results.

2.4 Conclusions

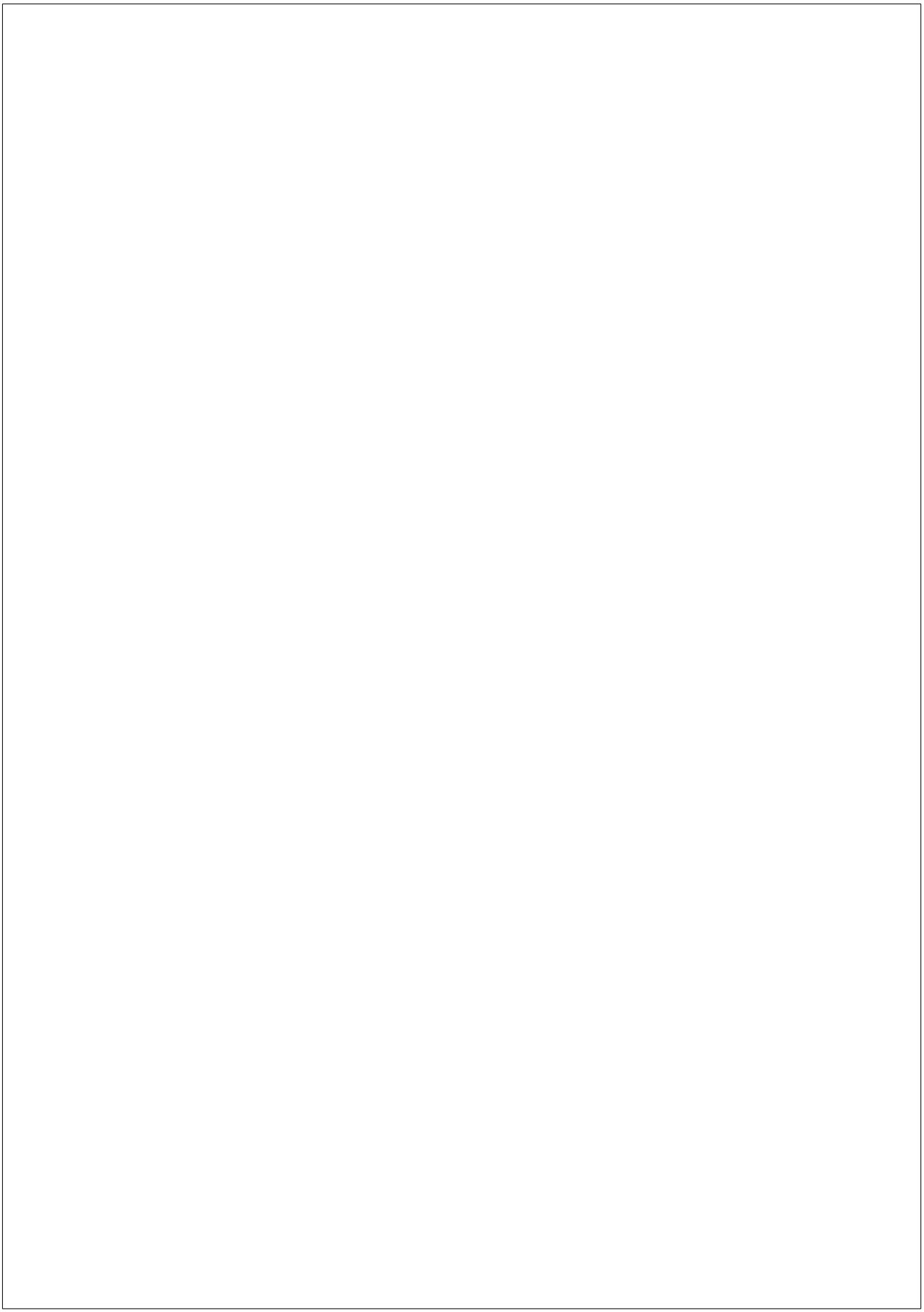
In this chapter, we have provided a formal definition of the deep learning architectures that we use in our approaches to the task of relative music loudness estimation, and the evaluation metrics that are typically used in music detection related tasks. We have also described and analyzed

both the public and private datasets that are available for these tasks, and we have provided an overview of the audio annotation tools that can be used to generate annotations about the presence of music. After this, we have conducted a literature review about previous algorithms for music detection related tasks.

Throughout the sections, we have observed the need for new data annotated for music detection related tasks that can fix the shortcomings of current datasets. A new dataset should comprise as many different broadcast scenarios as possible containing music that is both isolated and mixed with non-music sounds. These non-music sounds should include but not be limited to speech. Furthermore, a significant number of the audio files in the datasets should contain more than one class. Note that many authors define their taxonomy based on the content of their evaluation dataset; thus, poor datasets in terms of combinations of music and non-music sounds can lead to taxonomy-task combinations that represent a rather small part of the diversity of broadcast audio. We have also highlighted that incorporating information about the relative loudness of music in a dataset's annotations can be very useful during error analysis, as it allows to differentiate between the performance of an algorithm for foreground and background music. This information is obviously essential if the dataset is to be used for the task of relative music loudness estimation. These conclusions have led to the creation of the OpenBMAT dataset, which we detail in Section 4.2.

With regards to the reviewed algorithms, we consider that deep learning is the path to follow to keep improving BMAT's relative music loudness estimation technology. As an extremely successful architecture in this and many other fields, CNNs are the best option. In Chapter 5,

we describe the deep learning architectures that we develop throughout this thesis for the relative music loudness estimation task. We make use CNNS, TCNs, which is a CNN with the additional capacity of modeling sequential data, and experiment with the combination of CNNs and TCNs, proving that it produces a boost in performance as the combination of CNNs and RNNs do.



Chapter 3

DEVELOPMENT OF AN ANNOTATION TOOL

In this chapter, we present BMAT’s Annotation Tool (BAT), an open-source, web-based tool for the manual annotation of audio events that focuses on the annotation of the partial loudness of these events when two or more of them appear simultaneously. BAT is the topic of our first publication (Meléndez-Catalán et al., 2017), which we presented at the 3rd Web Audio Conference, held at the Centre for Digital Music of the Queen Mary University of London.

Prior to the creation of BAT, BMAT used a web-based annotation tool prototype that was not designed to annotate events, but rather to assign one or more labels to entire ten-seconds audio excerpts. This tool included a single annotation taxonomy that could comprehend an arbitrary number of classes. We used four of them: *Music*, *Speech*, *Background Music* and *Sound Effects*. It had several strengths: it was easy to use, the annotation process was fast, and it was web-based. As a web-based

tool it ran in a server where annotators could use it remotely without having to download, install or configure anything. This is very useful when working with crowd-sourcing or freelance platforms. Finally, any new annotation or modification was automatically ingested into a centralized database and ready to be used for evaluation without the need for any external ground truth files. However, this annotation method did not allow for the correct annotation of excerpts including class changes, which greatly limited the content that we could use for training. This was the main motivation behind the decision to create BAT.

The contents of this chapter are structured as follows: in Section 3.1, we list the design requirements of this new annotation tool. Then, in Section 3.2, we provide all the details about BAT and its development: we explain how we meet the design requirements, we offer an overview of the development process, we detail the structure of BAT’s database and the annotation process, we describe the different parts of the annotation interface as well as useful functionalities and shortcuts that facilitate annotation, and we show how BAT has changed since its initial integration into BMAT. After this, in Section 3.3, we present an experiment to check how reliable the partial loudness annotation system that BAT incorporates is. We close the chapter with the conclusions in Section 3.4.

3.1 Design requirements

Our new annotation tool should solve the shortcomings of the previous one, as well as preserve its advantages and provide new useful functionalities. We define the following requirements for the development of BAT:

1. Its annotation mechanism should allow for the precise annotation, over an audio’s waveform, of the start and end times of audio events. This solves the annotation precision issue of the previous tool.

2. It should permit the annotation of overlapping audio events and their partial loudness in the overlapping time interval. This makes the tool suitable for the relative music loudness estimation task, and also appropriate for both segmentation and detection tasks.

3. It should allow for the definition of multiple, independent taxonomies, so that it can be used for any segmentation or detection task.

4. It should enable the cross-annotation of audio. Cross-annotating is a way to validate the reliability of the annotations.

5. It should be open-source. This tool represents one of our contributions to the research community, and thus, we want to make it available to everyone.

6. It should be easy to use and have a clear annotation environment with no superfluous functionalities.

7. It should be web-based and easy to deploy in servers. As mentioned above, this facilitates its usage in crowd-sourcing and freelance platforms avoiding any downloads, and any installation and configuration processes.

8. It should include a database that provides an amenable way to store and manage annotations.

In Section 2.2.2, we provided an overview of several tools that have been used for the annotation of audio in the last two decades. Out of all of these, only audio-annotator (Cartwright et al., 2017) is web-based (req. 7). This tool fulfils several more of our requirements: it is open-source (req. 5), it allows for the annotation of audio events through the selection of time intervals over the waveform of an audio file (req. 1), and it has simple and attractive aesthetics (req. 6). However, audio-annotator is not a complete tool, but rather a web front-end interface that needs to be attached to a back-end that can store and manage the annotations. In the following section, we provide details about how we incorporate parts of audio-annotator into BAT, and the technologies behind the development of its back-end.

3.2 Development

In this section, we first specify the technologies that we use to create BAT’s front- and back-end. We then detail the structure of the database while explaining the annotation process and, after this, we describe the different parts of the annotation interface. Finally, we list the modifications that BAT has undergone in its integration into BMAT.

3.2.1 Technologies

The specific parts of audio-annotator that we use in the front-end of BAT are its extended version of wavesurfer.js and some aesthetic details of their design such as the taxonomy presentation. We show a screenshot of this tool in Fig. 3.1. The original version of wavesurfer.js allows for the playback of audio, the visualization of its waveform, and the selection of

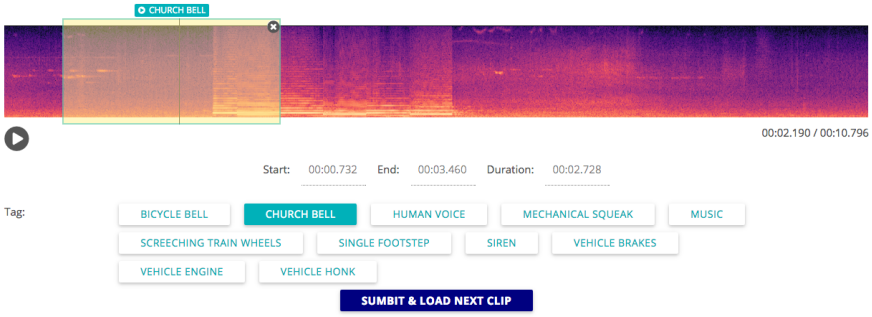


Figure 3.1: Screenshot of audio-annotator’s front-end annotation interface.

time intervals over it in web browsers using JavaScript, HTML5 and the Web Audio API[1]. The extended version incorporates class labels to the time intervals, and the possibility to switch between the visualization of the audio’s waveform and its spectrogram. To create BAT’s front-end we assemble these parts with our own HTML5 templates and we add several other functionalities that enable the annotation of the partial loudness of overlapping audio events (req. 2), and facilitate the annotation process.

We implement BAT’s back-end using Django,[2] a Python framework for the development of web applications. In the back-end, we define a set of object models that structure a PostgreSQL[3] database (req. 8) where all the information related to the annotation process is stored. We also define a set of functions that improve the ways a user can interact with this database. Django offers the possibility of accessing and modifying the contents of the database through the admin web-page, and also externally

[1] https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[2] https://www.djangoproject.com/

[3] https://www.postgresql.org

using the Ipython shell[4] or Jupyter notebooks[5]. The Django web application and the database run in separate Docker containers that we link together using Docker Compose. Using docker facilitates the deployment of BAT in servers (req. 7).

BAT accepts audio in WAV format and works, at least, with Mozilla Firefox, Google Chrome and Opera. It can be found at its github repository[6] with a GNU AGPL v3.0 license. The repository contains the source code, documentation about the installation and the annotation processes, and the necessary docker files.

3.2.2 Models and annotation process

In Django we design models that define the structure of the database. We, then, populate this database through the creation of objects of these models. Fig. 3.2 shows the relationship between the models in BAT's database. In this section, we present these models as we detail the annotation process.

The first step to start annotating audio with BAT is to create a project (*Project* model). Projects are the framework that links a taxonomy, i.e., a set of classes (*Class* model), to the set of audio files to be annotated using this taxonomy (*Wav* model). We can assign an unlimited number of classes to a project, which ensures taxonomic flexibility (req. 3).

The audio files uploaded to a project are automatically split into segments of a fixed length (*Segment* model). This length must be specified before uploading the audio files. We have decided to work with fixed

[4] `ipython.org`
[5] `https://jupyter.org`
[6] `https://github.com/BlaiMelendezCatalan/BAT`

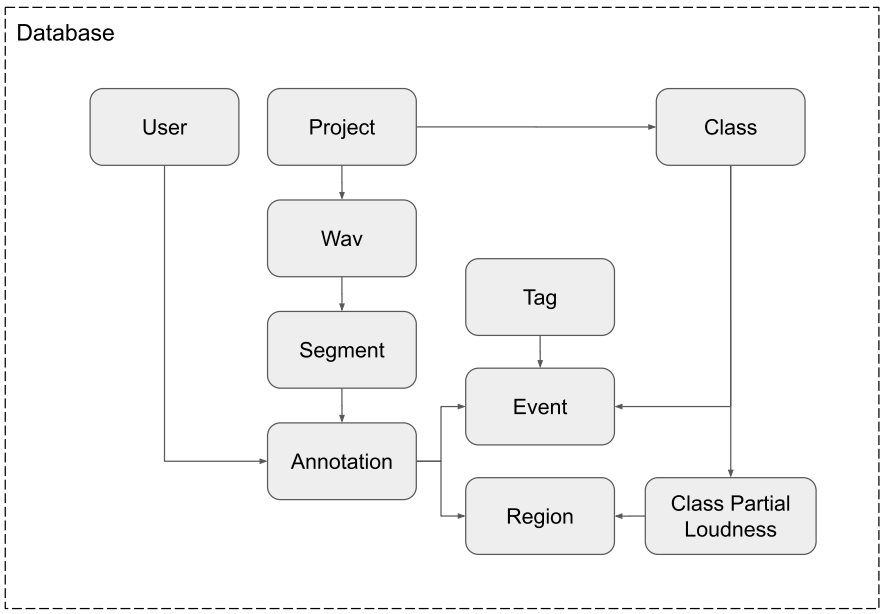


Figure 3.2: Relationship between BAT’s database models.

length segments instead of the full audio files because this allows for an annotation with constant time resolution and avoids actions such as zooming and scrolling, which introduce a cost in time and increase the complexity of the annotation process. Additionally, this setup favors the usage of crowd-sourcing strategies by allowing the distribution of the workload among several annotators. Nevertheless, it is also possible to generate only one segment per audio file for tasks such as global key estimation, genre recognition, etc. Note that these segments have nothing to do with the segments that appear in Section 2.1.2.1 when describing the segment-based evaluation method.

In BAT, each user (*User* model) has one of two roles based on the tasks they perform throughout the annotation process: the administrator and the annotator. The administrator is the user that creates projects, defines taxonomies, uploads the audio data and chooses the length of the segments. It can also visualize all the existing annotations, but not modify or generate them. The other type of user, the annotator, has access only to its own annotations, but can modify them or generate new ones as long as there are still non-annotated segments. To annotate the content of an audio file for some tasks, the annotator might have to incorporate a certain amount of subjectivity. In these cases, the tool is prepared to manage the annotations of more than one user, i.e, to allow for cross-annotation (req. 4). Having multiple annotations is a way to validate the reliability of the annotations and to detect the parts that produce more disagreement between annotators.

When a user selects a segment to annotate it, BAT creates an annotation (*Annotation* model), which is uniquely linked to both the segment and the user. BAT divides the annotation process into two sequential phases: the event identification phase and the partial loudness annotation phase. In the event identification phase, the annotator should select the time intervals of the waveform containing the audio events that are relevant to the current task, for instance, a person speaking, a certain chord, a note of a soloing saxophone, etc. Each selection creates an event (*Event* model). When we create the project, we decide if we allow events to overlap in time. We would enable it for detection tasks such as instrument detection, and disable it for segmentation tasks such as chord recognition. The annotator must assign a class from the linked taxonomy to every event and optionally add a tag (*Tag* model) to them. Tags allow to further describe

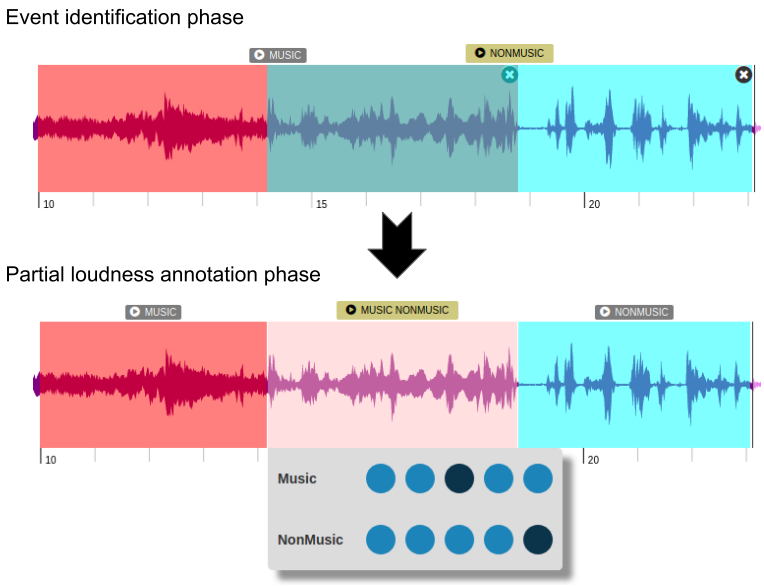


Figure 3.3: (Top) Event identification phase: the annotator creates two events over the waveform. (Bottom) Partial loudness annotation phase: the annotator assigns a partial loudness value to each event where they overlap.

the selected audio event. There is no restriction in the number of tags that the annotator can use.

If some of the events created in the event identification phase overlap in time, it is mandatory to go through the partial loudness annotation phase. In this second phase, non-overlapping events directly become regions (*Region* model). Otherwise, when two or more events overlap, BAT creates a region for each time interval with different classes as shown in Fig. 3.3. If a region contains more than one class, the annotator needs to assign a partial loudness value to each of these classes (*Partial Loudness*

model). This value has five possible levels represented by integers in a range from 1 to 5. The class or classes with the highest partial loudness in the region must always receive a partial loudness value of 5. The contribution to the total loudness of the highest partial loudness is taken as reference to assign partial loudness values to the other classes. A relative annotation method prevents inconsistencies due to the volume at which the annotator listens to the audio. If a region has only one class, it automatically receives a partial loudness value of 5.

BAT offers the possibility of switching from one annotation phase to the other; however, all partial loudness information is lost if the annotator goes back to the event identification phase, as events might be modified, and thus, produce different regions in the partial loudness annotation phase. During the event identification phase, the annotator cannot finish the annotation or access the partial loudness annotation phase if there are events with no assigned class. Similarly, in the partial loudness annotation phase, it is not possible to finish the annotation if there are unassigned loudness values. The tool will display a warning with the corresponding explanation every time the annotator incurs in one of these violations.

3.2.3 Annotation interface

To start annotating, the annotator needs only to select the name of a project and BAT will automatically deliver all the corresponding segments sequentially using a simple and clear interface that displays only the most essential elements in a balanced and colorful way. We show this interface in Fig. 3.4. The interface is organized in rows of elements: the first row (1) contains two buttons that expand a text box with either annotation tips or a list of the annotation shortcuts. In the next row (2), at the left side

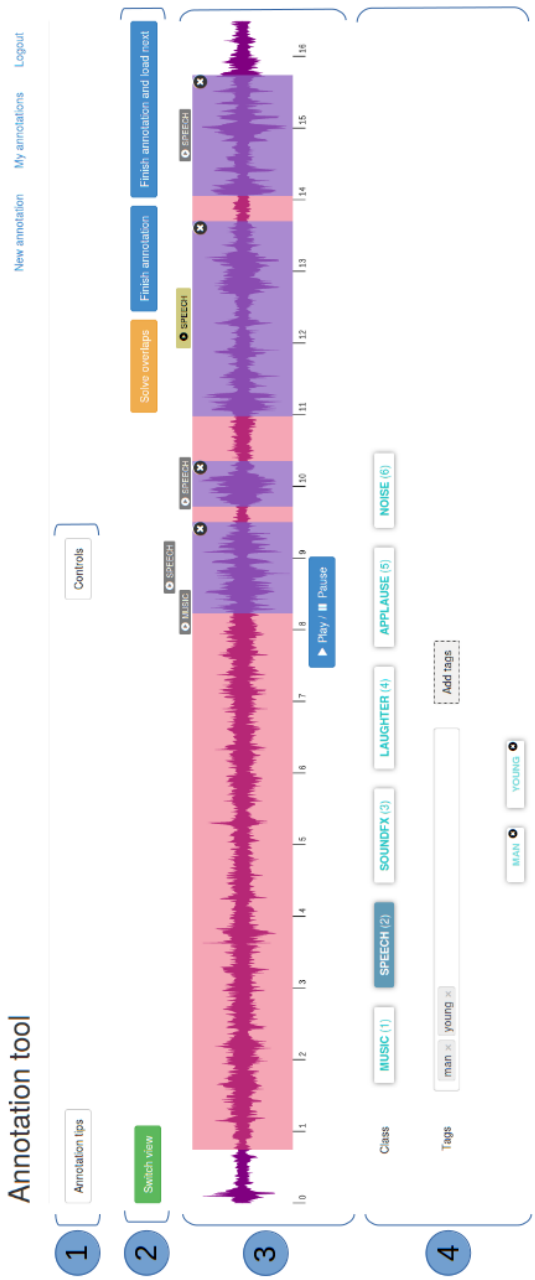


Figure 3.4: BAT’s annotation interface.

we find the button to switch the audio visualization from waveform to spectrogram and vice versa, and at the right side, the button to transition between the two phases of the annotation process, and the buttons to finish the current annotation. After this, (3) we have the audio visualization with a button to play and pause its playback. The annotator can create events over the waveform by just clicking and dragging. The last row (4) shows all the classes in the taxonomy and highlights the class/es of the selected region or event. It also shows the tags assigned to it.

BAT also integrates several functionalities and keyboard shortcuts that facilitates the annotation. Using the keyboard, the annotators can set the class of the regions, play and pause the audio, and expand the limits of a region to the boundaries of either another region or the entire segment. The functionalities include: the prevention of overlaps if the project does not allow them, the unification of the limits of two events, if they are very close to each other in order to avoid the creation of small overlaps or gaps between them, and the deletion of small events when created accidentally by dragging the mouse after a click.

3.2.4 Integration into BMAT

BAT has undergone several modifications since its introduction inside BMAT's ecosystem either to add new features to it, to fix bugs, or to refactor its code. Some of the most important changes are:

- The incorporation of its database models as the annotation application of a larger Django framework that also includes applications for evaluation and training data generation.
- The introduction of two new database models: the *WavsCollection*

models, which binds audio files together in a supraproject manner, and the *ClassInstance* models, which links projects and classes, fixing an error of the original design that forced the administrator user to create the same class for every project where it was used.

- The elimination of the duplicity of the *Event* and *Region* models: currently, events are deleted and replaced by regions when advancing from the event identification phase to the partial loudness annotation phase, and reenacted if the annotator needs to go back to the event identification phase. Previously, events were only replaced but not removed.

- The possibility for an administrator user to modify the annotations of other users.

- The creation of functions to export annotations to a ground truth CSV file, or to compute the agreement between annotators over a set of segments.

- The incorporation of new annotation modes where segments are presented in random order or where only one annotator is allowed to annotate a particular segment. Both modes are useful for crowdsource annotation.

- The possibility to upload and play the video and audio parts of an MP4 audio-visual file. In some tasks, the visualization of the source of a sound can be very informative.

3.3 Evaluation experiment

The most distinguishing feature of BAT with respect to other tools is that it offers a mechanism to annotate the partial loudness of an audio event in the presence of other simultaneous events. In this experiment, we assess the inter-annotator agreement in the annotation of partial loudness. We consider that a high inter-annotator agreement indicates that the system can reliably provide the requested information. For the rest of this thesis, we will just use the term agreement when referring to inter-annotator agreement.

3.3.1 Evaluation methodology

We deliver several annotated regions including more than one class to the annotators and ask them to introduce their partial loudness values. In this way, we can specifically evaluate the agreement in the annotation of partial loudness avoiding errors coming from the event identification phase.

To carry out the experiment we gather four annotators. Before starting the experiment, we describe the annotation process, as well as the available controls, and allow them to annotate a few examples until they feel confident using BAT. The dataset used for the evaluation contains eight recordings with a duration of one minute. Each of these recordings belong to one of the following eight types of broadcast media programs: children programs, documentaries, entertainment programs, music programs, news broadcasts, series and films, sport programs, and talk shows. We load them to the database with a segment length of 30 seconds producing a total of 16 segments.

Coinciding annotators	Ocurrences	Accumulated percentage	Averaged agreement
4 (100%)	34 (49.28%)	49.28%	83.33%
3 (75%)	24 (34.78%)	84.06%	
2 (50%)	11 (15.94%)	100%	
0 (0%)	0 (0%)	100%	

Table 3.1: Results of the experiment to validate BAT’s partial loudness annotation mechanism.

3.3.2 Evaluation results

As shown int Table 3.1, the values assigned by all the annotators are identical for 49.28% of the regions. This percentage rises to 84.06% if we consider the regions where three of the annotators coincide and to 100% for the case of two annotators. This means that there is no region that has been annotated differently by all annotators. We compute the average agreement as the average of the percentage of occurrences weighted by the percentage of coinciding annotators. We reach an average agreement of 83.33%.

3.4 Conclusions

In this chapter, we have described BAT: an open-source, web-based, tool for the annotation of audio events that can also provide information about their partial loudness in a reliable way. BAT allows the definition of different taxonomies to adapt to multiple tasks and offers the possibility to cross-annotate audio data. It is also easy to deploy in servers thanks to the usage of Docker. BAT has proven to be functional and very useful to BMAT, providing annotations for hundreds of hours of audio. It is also

in constant evolution to improve the annotation process. In the context of this thesis, we use it to annotate two datasets that are essential in the development of our computational approaches to the task of relative music loudness estimation. We describe these datasets in the next chapter.

Beyond the scope of this thesis, we have recently annotated a new dataset using BAT that includes only stereo audio. This dataset could lead to important future advances in the relative music loudness estimation technology. We used this dataset as the evaluation dataset for the music detection and the relative music loudness estimation tasks of MIREX 2019. Using data annotated with BAT, we have also trained algorithms for other tasks such as the detection of audience noise, or an estimator of how audible music is in an audio recording. Furthermore, we have used BAT in several one-time projects related to the study of the amount of music played on national and international radio stations and TV channels.

Chapter 4

COMPILATION AND ANNOTATION OF DATASETS

In Section 2.2.1.2, we have highlighted the need for new annotated quality datasets that are specific to tasks related to the detection of music. In this chapter, we describe the creation of two such datasets: one private and the other public. The private dataset constitutes a contribution to BMAT, while the public dataset, which we have called OpenBMAT, represents a contribution to the research community, and led to the publication of our second paper (Meléndez-Catalán et al., 2019), in the transactions of the international society for music information retrieval (TISMIR) journal.[1]

The creation of these two datasets is an essential step towards the realization of the goals of this thesis described in Section 1.3. We build the private dataset to train the Deep Music Detector: the algorithm for the task of relative music loudness estimation that BMAT is currently using

[1] https://transactions.ismir.net/articles/10.5334/tismir.29/

in production. With the public dataset, we pursue the introduction of this task in the research field of MIR, and the promotion of transparent, comparable and reproducible research. In this respect, we use it to train and evaluate novel computational approaches to the task of relative music loudness estimation, which we describe in Chapter 5.

This chapter is structured as follows: Section 4.1 and Section 4.2) are devoted to the description of the private dataset and OpenBMAT, respectively. In these sections, we present the audio content of the datasets, and the process to annotate it, as well as its distribution by class. In the case of OpenBMAT, given that it has been cross-annotated by three annotators, we are also able to validate the reliability of its annotations. Finally, in Section 4.3 we provide the conclusions of the chapter.

4.1 Private dataset

This section includes a description of the audio content of the private dataset, as well as the annotation methodology used to annotate it, and the distribution of its content by class.

4.1.1 Raw corpus

This dataset contains approximately 44 hours of audio divided in 1322 two-minute audio files. We consider that having many short audio files allows the dataset to include a greater variety of contexts. Nevertheless, these audio files are long enough to include class changes, and to provide a significant amount of information about the temporal evolution of the audio. Each of them comes from a different recording that we randomly

sample from BMAT’s private database. These recordings contain audio broadcast by TV channels and radio stations from all over the world.

4.1.2 Annotation methodology

For the annotation of the private dataset, we inherited the annotation taxonomy that we used with the previous annotation tool. This taxonomy includes the classes *Music*, *Speech*, *Sound Effects*, and *Audience*. The annotation using BAT allows audio events of these classes to overlap. If that happens, we need to assign partial loudness values to the overlapping audio events. This annotation is carried out by a single annotator, at an annotation speed of approximately 5.5 hours to annotate an hour of audio.

In order to use this dataset for the task of relative music loudness estimation, we have to map every possible combination of audio events and their partial loudness into the classes *Foreground Music*, *Background Music*, and *No Music*. The mapping consists of the following three conditions, where *m*, *s*, *sfx*, and *a* represent the partial loudness value of the audio events of class *Music*, *Speech*, *Sound Effects*, and *Audience*, respectively.

```
1 if (m == 5 and
2   (0 <= s <= 3 or 0 <= sfx <= 4 or 0 <= a <= 4)):
3     # Map to Foreground music
4 elif m >= 1 and (s >= 4 or sfx == 5 or a == 5):
5     # Map to Background music
6 elif m == 0 and (s == 5 or a == 5 or sfx == 5):
7     # Map to No music
```

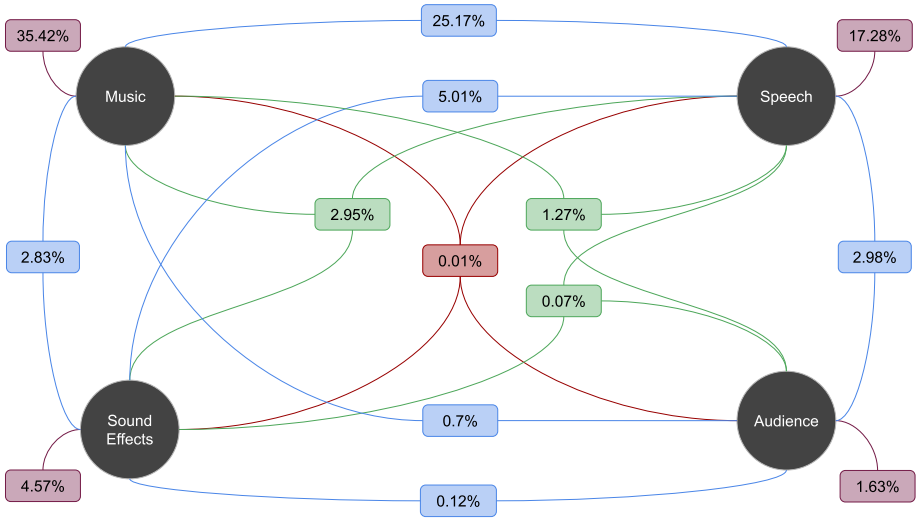


Figure 4.1: Percentage of the total duration corresponding to every existing combination of the classes of the annotation taxonomy.

To obtain a ground truth that is suitable for the task of relative music loudness estimation, we apply these conditions to every region (an object of the *Region* model described in Section 3.2.2) in the annotations. Note that any class that is not present in the region is assigned a partial loudness value of 0.

4.1.3 Content distribution

Table 4.1 shows the percentage of the total content annotated as each of the classes of the annotation taxonomy. The sum of the percentages surpasses 100% because classes overlap. Over two thirds of the dataset contain the class *Music*, more than half of it contains the class *Speech*, and around 22% of it contains other non-music sounds of which approxi-

Classes	Music	Speech	Sound Effects	Audience
Content	68.35%	54.72%	15.55%	6.77%

Table 4.1: Percentage of the total content annotated as each of the classes of the annotation taxonomy.

	Foreground Music	Background Music	No Music
Content	37.16%	31.19%	31.65%

Table 4.2: Percentage of the total content mapped to each of the classes of the relative music loudness estimation taxonomy.

mately 7 percentage points are audience noises. The percentage of audio files that include class changes is 85%. Fig. 4.1 shows the percentage of the total content that corresponds to every existing combination of classes. After applying the mapping conditions, we obtain a fairly balanced distribution of content with 37% of *Foreground Music*, 31% of *Background Music*, and 32% of *No Music*, as shown in Table 4.2. The percentage of audio files that include class changes using these three classes is 72%.

4.2 Public dataset: OpenBMAT

Public annotated data is essential for the development of the field of MIR as it promotes transparency in research and provides a solid evaluation framework with which we can fairly compare algorithms and keep track of the evolution of tasks. On top of that, OpenBMAT solves the shortcomings of the datasets described in Section 2.2.1.2. OpenBMAT is publicly available under request for non-profit purposes at Zenodo[2].

[2] https://zenodo.org/record/3381249

4.2.1 Raw corpus

OpenBMAT contains 27.4 hours of audio divided in 1647 one-minute audio files. Each of these audio files comes from a different recording that we have sampled from BMAT's private database. As mentioned for the private dataset, we consider that having many short audio files allows the dataset to include a richer variety of contexts. In this case, we reduce the duration of the audio files from two minutes to one minute. In this way, we obtain a larger number of contexts, while the audio files are still long enough to include class changes, and to provide enough information about the temporal evolution of the audio.

We have forced the sampled audio files to cover a set of varied program types to ensure that the dataset is representative of several different broadcast contexts. The selected program types are: children programs, documentaries, entertainment programs, music programs, news broadcasts, series and films, sport programs, and talk shows. Including such a variety of program types guarantees the presence of music that is both isolated and mixed with different types of non-music sounds.

Unfortunately, in BMAT's database only the recordings from certain countries include information about the program type. This has constrained the dataset to audio broadcast by TV channels in France, Germany, Spain and the United Kingdom. We set a limit of 60 audio files for each country and program type, but for several combinations there were not enough audio files to reach that value. Fig. 4.2 shows the distribution of audio files by program type for each country. All the audio files in the dataset are 16-bit monophonic WAV files at a sampling rate of 22050 Hz and have been extracted from audio broadcast during 2017.

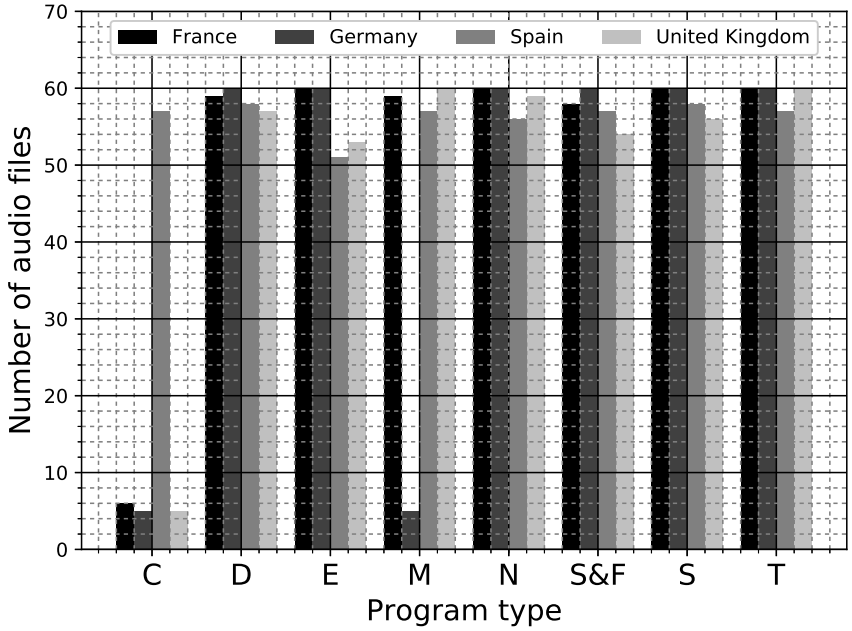


Figure 4.2: Distribution of audio files by program type and country. The program types are: children programs (C), documentaries (D), entertainment programs (E), music programs (M), news broadcasts (N), series & films (S&F), sport programs (S) and talk shows (T).

4.2.2 Annotation methodology

OpenBMAT has been manually cross-annotated by three different annotators: two males and one female of ages comprised between 20 and 40 years, and experience working with sound and/or music. Cross-annotating allows us to assess the reliability of the annotations that are produced through the study of the inter-annotator agreement (see Section 4.2.3.2). The reliability of the annotations is our main priority for the annotation of OpenBMAT. In order to increase this, we modify some as-

pects of the annotation methodology with respect to the annotation of the private dataset.

The methodology that we use to annotate the private dataset makes it possible to distinguish between different types of foreground and background music depending on the kind of non-music that appears mixed with the music. Even though this information can be useful, especially for error analysis, we consider that it might lead to an increase of the disagreement between annotators, and thus, to a decrease in the reliability of the annotations. On top of this, we believe that the annotation time can be reduced by using a taxonomy that does not allow overlaps. A taxonomy with non-overlapping classes is also more in line with the nature of segmentation tasks such as the relative music loudness estimation task, requiring much simpler mapping conditions.

4.2.2.1 Taxonomy description

As a dataset for the tasks of relative music loudness estimation and music detection, the annotation taxonomy used in OpenBMAT needs to be compatible with the classes of these two tasks. The taxonomy of the relative music loudness estimation task contains three classes: *Foreground Music*, *Background Music* and *No Music*; and the taxonomy of the music detection task, if we understand it as a segmentation task, includes the classes *Music* and *No Music*. In including the classes of the relative music loudness estimation task in the annotation taxonomy, we are, of course, also incorporating those of the music detection task.

We consider, however, that a more fine-grained taxonomy can provide extra information without adding too much complexity to the annotation process: we separate the isolated music from the rest of foreground music

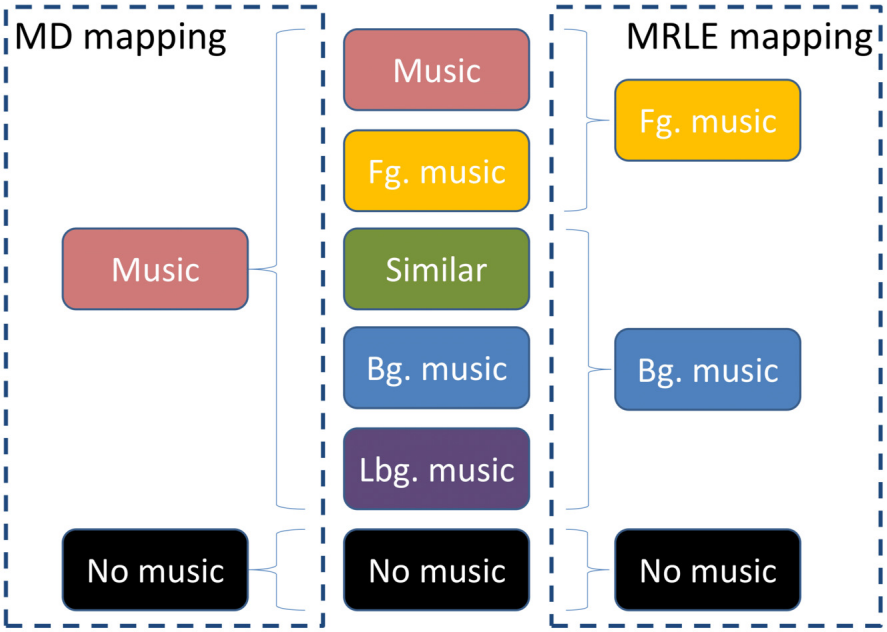


Figure 4.3: (Left) MD mapping: mapping from the annotation taxonomy to the music detection taxonomy. (Right) RMLE mapping: mapping from the annotation taxonomy to the relative music loudness estimation taxonomy.

creating the *Music* class, and we subdivide the background music in three classes according to the relative loudness of the music: *Similar*, *Background Music*, and *Low Background Music*. The *No Music* class remains as it is. In Fig. 4.3, we show the classes of the annotation taxonomy and the mappings corresponding to the relative music loudness estimation task (RMLE mapping) and the music detection task (MD mapping). The definition of the classes is the following:

- **Music**: isolated music.

- **Foreground Music**: mainly music with low-volume non-music in the background.
- **Similar**: music and non-music mixed at similar volumes.
- **Background Music**: mainly non-music with music in the background.
- **Low Background Music**: mainly non-music with music in the background at such a low volume that it is hard to hear.
- **No Music**: isolated non-music.

Notice three characteristics of the resulting annotation taxonomy: first, all of its classes are mutually exclusive, which is mandatory for segmentation tasks; second, with these classes we can annotate the whole duration of any audio; and third, we can separate them into *isolated* classes and *mixed* classes. *Music* and *No Music* are isolated classes because a region (an object of the *Region* model described in Section 3.2.2) of these classes can contain either music or non-music sounds but not a combination of both. The rest of the classes are mixed classes because regions annotated as such must contain both music and non-music sounds. We use this distinction in future sections.

4.2.2.2 Annotation process

The annotation mechanism consists of creating non-overlapping time intervals over the waveform of an audio file and assigning a class to each of them. The annotation mechanism might be simple, but we are asking the annotators to describe the relative loudness of music, which has a continuous nature and can have fast variations, using a discreet set of classes.

This process involves a notable level of subjectivity, and can lead to significantly different annotations. This is why we provide the annotators with a set of annotation steps:

1. Annotate all *No Music* time intervals that are longer than 1 second.
2. Group non-annotated, non-music sounds that are separated by less than 1 second and have similar loudness in comparison with the simultaneous music into the same time interval.
3. Iteratively annotate the resulting groups that are longer than 1 second as one of the mixed classes. If a group is shorter than 1 second:
   (a) and it is separated by less than 1 second form another group, merge them. The time interval takes the class that is majority.
   (b) and it is surrounded by *No Music*, annotate it as the appropriate class.
   (c) and it is surrounded by non-annotated audio, leave the group with no annotation.
4. Annotate any part of the audio that is not yet annotated as *Music*.

Before starting the annotation of the dataset, all three annotators have been trained to use BAT and have understood the annotation steps. Throughout the whole process they have been allowed to ask questions and we have regularly provided feedback. On average, they spent approximately 130 hours annotating the dataset. This means that the annotation speed was around 4.75 hours per hour of audio annotated, which is 14% faster than with the annotation methodology used for the private dataset.

4.2.3 Analysis of the annotations

In this section, we first provide statistics about the content of the annotations; and we then validate the annotation methodology through the calculation and analysis of the agreement between annotators.

4.2.3.1 Content distribution

After the cross-annotation process, we obtain three different annotations of the same content. Table 4.3 shows the content distribution as annotated by each annotator for the complete taxonomy and both mappings. We observe the strongest variance in percentage of content for the *Similar* and *Background Music* classes. We highlight the presence of around 4 to 5% of *Low Background Music*, which we consider to be one of the most challenging types of content.

Once we apply the RMLE mapping the percentages become very similar for all annotators. The part of the dataset annotated as *Foreground Music* is approximately 15%, while the part annotated as *Background Music* represents about 35% of its total duration. With the MD mapping, we find that approximately 50% of the dataset has no music and the other 50% has music either isolated or mixed with non-music sounds. Furthermore, if we divide the dataset in terms of isolated and mixed classes as explained in Section 4.2.2.1, the average proportion is around 59% and 41%, respectively. The dataset comes divided into 10 predefined splits that preserve the same content distribution. Finally, Table 4.4 shows the percentage of audio files that include class changes by annotator for the complete taxonomy and both mappings. The average is 68.3% for the complete taxonomy, 65.5% for the RMLE mapping, and 51.2% for the MD mapping.

All classes						
Annotator	**Music (%)**	**Fg. Music (%)**	**Similar (%)**	**Bg. Music (%)**	**Low Bg. Music (%)**	**No Music (%)**
Annotator 1	11.14	5.46	5.4	24.22	4.84	48.94
Annotator 2	7.82	4.88	13.5	19.64	4.14	50.02
Annotator 3	9.57	5.43	14.2	16.36	4.1	50.34
Relative music loudness estimation classes						
Annotator	**Fg. Music (%)**		**Bg. Music (%)**			**No Music (%)**
Annotator 1	16.6		34.45			48.94
Annotator 2	12.7		37.28			50.02
Annotator 3	15		34.66			50.34
Music detection classes						
Annotator	**Music (%)**					**No Music (%)**
Annotator 1	51.05					48.94
Annotator 2	49.98					50.02
Annotator 3	49.66					50.34

Table 4.3: Percentage of audio annotated by each annotator as the classes of the complete taxonomy, and the classes of the relative music loudness estimation and music detection tasks.

Annotator	All classes	RMLE classes	MD classes
Annotator 1	67.09%	65.76%	50.09%
Annotator 2	70.19%	66.55%	53.19%
Annotator 3	67.64%	64.24%	50.33%

Table 4.4: Percentage of audio files with class changes by annotator for the complete taxonomy and both mappings.

4.2.3.2 Inter-annotator agreement

As explained in Section 4.2.2, we cross-annotate the dataset to allow for the assessment of the reliability of the produced annotations. We consider that obtaining reliable annotations validates the definition of the taxonomy and the annotation process ensuring the usability of the dataset. The information that we use for this assessment is the percentage of agreement between the three annotators in the annotated classes.

We define two different levels of agreement: full agreement, which happens when all three annotators have annotated the same class; and partial agreement, which happens when at least two annotators have annotated the same class. We compute the percentage of full agreement $\%FA_{af}$ and partial agreement $\%PA_{af}$ in an audio file as the time during which the agreement level is reached (t_{FA} and t_{PA}, respectively) divided by the duration of the audio file T_{af} as shown in eq. 4.1 and eq. 4.2. To obtain the percentage of full agreement $\%FA$ and partial agreement $\%PA$ for the whole dataset, we compute the mean for all N audio files as shown in eq. 4.3 and eq. 4.4. These values can be computed considering all the classes in the taxonomy, but also for the relative music loudness estimation and the music detection classes.

Agreement level	**All classes Agr (%)**	**RMLE classes Agr (%)**	**MD classes Agr (%)**
%FA	68.18%	89.1%	94.78%
%PA	96.75%	99.79%	100%
%PW (annot. 1 & 2)	77.46%	91.7%	96.22%
%PW (annot. 2 & 3)	76.97%	92.78%	96.78%
%PW (annot. 1 & 3)	78.66%	93.52%	96.55%

Table 4.5: Percentages of full, partial and pair-wise (PW) agreement (Agr) for the whole dataset. These values have been computed for the complete taxonomy and both mappings.

$$\%FA_{af} = \frac{t_{FA}}{T_{af}} \tag{4.1}$$

$$\%PA_{af} = \frac{t_{PA}}{T_{af}} \tag{4.2}$$

$$\%FA = \frac{1}{N}\sum_{n=1}^{N} \%FA_{af}(n) \tag{4.3}$$

$$\%PA = \frac{1}{N}\sum_{n=1}^{N} \%PA_{af}(n) \tag{4.4}$$

Table 4.5 shows the $\%FA$ and $\%PA$ when considering all classes as well as when applying both mappings. It also presents these agreement percentage values for each pair of annotators, i.e., the percentage of pair-wise agreement $\%PW$. We observe that when considering all classes, there is already a $\%PA$ of 96.75%. This percentage increases to 99.79% when applying the RMLE mapping. We also observe that the $\%FA$ con-

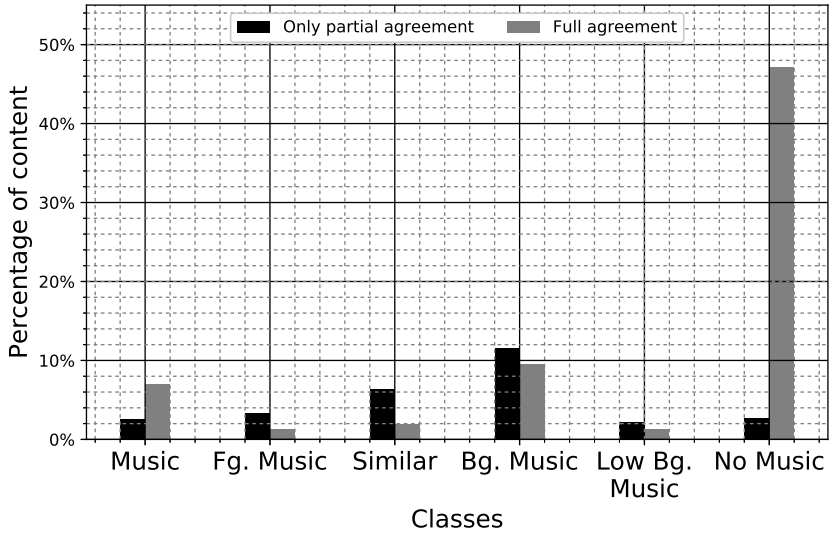


Figure 4.4: Percentage of the content of OpenBMAT by class and agreement level.

sidering all classes is 68.18%. Fig. 4.4 reveals that most of this agreement comes from isolated classes –especially the *No Music* class– as in all mixed classes there is a higher percentage of partial agreement than full agreement. The $\%FA$ increases to 89.1% when applying the RMLE mapping, and to 94.78% when applying the MD mapping. Fig. 4.5 provides insight on the distribution of full agreement among audio files. It shows the percentage of audio files with a $\%FA_{af}$ over a certain value when using the RMLE mapping. We observe, for instance, that over 35% of the audio files have a $\%FA_{af}$ higher than 99% and that almost 90% of the audio files have a $\%FA_{af}$ higher 70%.

Fig. 4.6 presents the percentage of the content with full (diagonal) or partial agreement for each class divided by the classification of the third

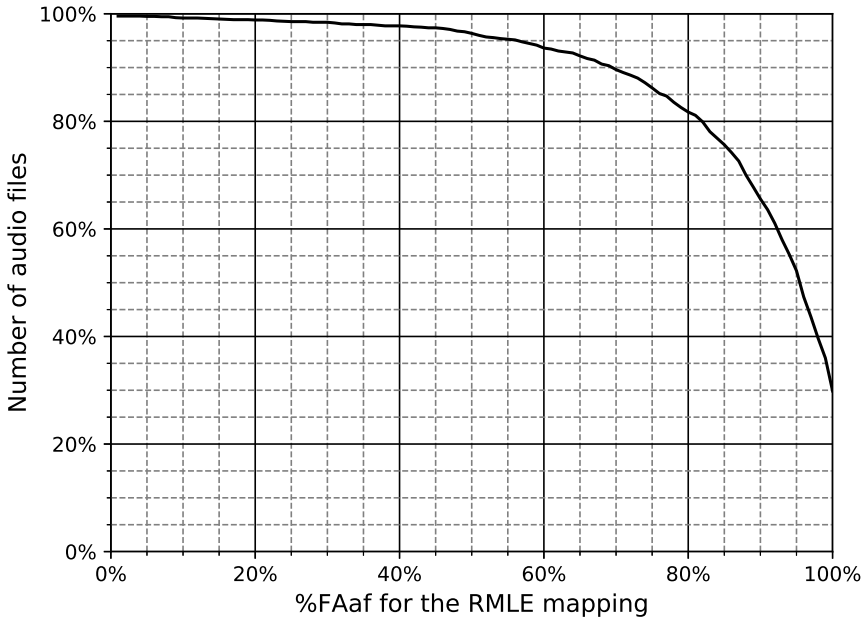


Figure 4.5: Percentage of audio files accumulated over a certain $\%FA_{af}$ value using the relative music loudness estimation classes.

annotator. From this figure we extract the two main sources of partial agreement when considering all the classes in the taxonomy. The most common source is for one of the annotators to select an adjacent class in terms of loudness. This affects all classes to a different extent except for the *No Music* class. The second source of partial agreement appears when one of the annotators is unable to detect the music due to its low volume and annotates *No Music* instead of *Low Background Music*.

When no annotator coincides in the annotation of a time interval, we consider that there is disagreement. Considering all classes in the taxon-

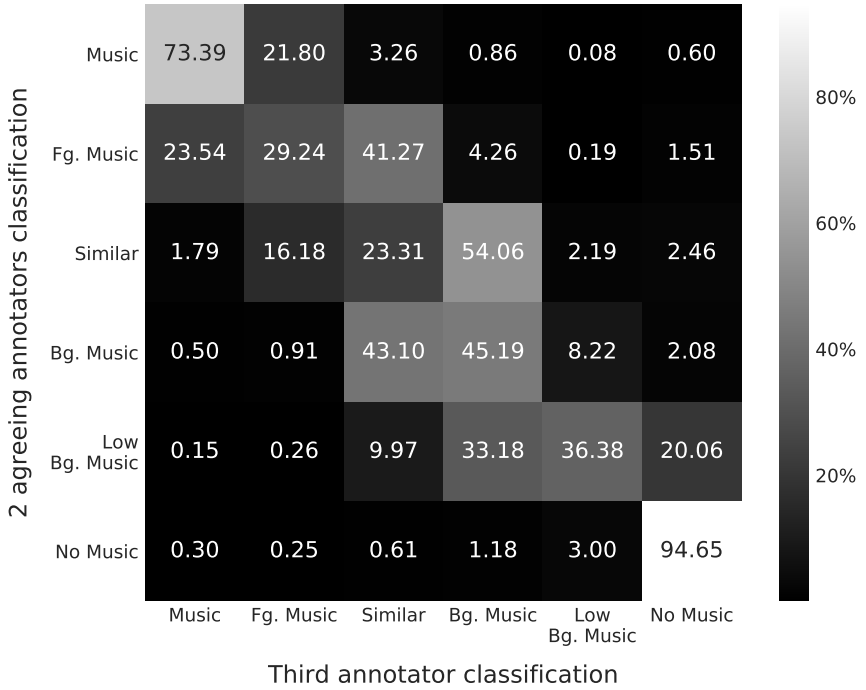


Figure 4.6: (Rows) Class annotated by two annotators. (Columns) Class annotated by the third annotator. (Values) Percentage of the content with full (diagonal) or partial agreement for each class divided by the classification of the third annotator.

omy, there are two main sources of disagreement: the first one takes place when annotators disagree between *No Music* and any of the other classes due to a different interpretation of what music is. Examples of this can be background tones, musical sound effects such as a church bell or a ringing phone or even experimental music or music of uncommon styles. The second source of disagreement appears due to a different interpretation of what components of the audio belong to the music. This happens, for in-

stance, when the audience claps following the beats of the music or when there are soft noises overlapping with the music that some annotators may find irrelevant. This can lead to strong differences in the annotation and represents a significant percentage of the parts of the dataset with disagreement.

4.2.4 Metadata and Storage

Metadata is all the information related to the audio files in OpenBMAT. There are two sources of metadata per audio file: the annotations and the agreement. The annotations include the three original annotations as well as their MD and RMLE mappings. In total, there are nine annotations per audio file. The agreement metadata contains the $\%FA_{af}$ and the $\%PA_{af}$ of each audio file for the original annotations and both mappings, and the set of time intervals with either full or partial agreement. These time intervals give potential users the possibility to train or test their algorithms using, for instance, only the subset with full agreement. All this metadata is provided in JSON format. The metadata also includes the annotations in two additional formats: (1) as exported from BAT (one CSV file for each annotator) and (2) as separate TSV files for each audio file, annotator and taxonomy.

Apart from the metadata and the audio, OpenBMAT also contains:

- 10 predefined splits to allow for K-fold cross-validation. We have randomly assigned each audio file to a subset and the resulting subsets preserve the original proportion of the relative music loudness estimation and music detection classes.
- A python module with utilities such as (1) the generation of the an-

notations in JSON and TSV formats from BAT annotations, (2) the generation of the full and partial agreement audio subsets and (3) the possibility of loading the annotations using the open evaluation library used in the detection and classification of acoustic scenes and events (DCASE) challenge.[3] The DCASE challenge is an international competition similar to MIREX, but exclusive to scene and event analysis methods.

- A README file including a general description of the dataset and all the details about its structure and contents.

4.3 Conclusions

In this chapter, we have described two datasets containing audio extracted from BMAT's private database that we have annotated using BAT. The first dataset is private and contains approximately 44 hours of audio divided in 1322 two-minutes audio files obtained from radio stations and TV channels from all over the world. The dataset has been annotated by a single annotator and the annotation includes information about the presence of music, speech, sound effects and audience noises, along with their partial loudness in case of overlap. More than two thirds of the dataset contains music, approximately half of it contains speech, and around 22% of the dataset contains other non-music sounds including 7 percentage points of audience noises. Out of all the audio files in the dataset, 85% contain class changes.

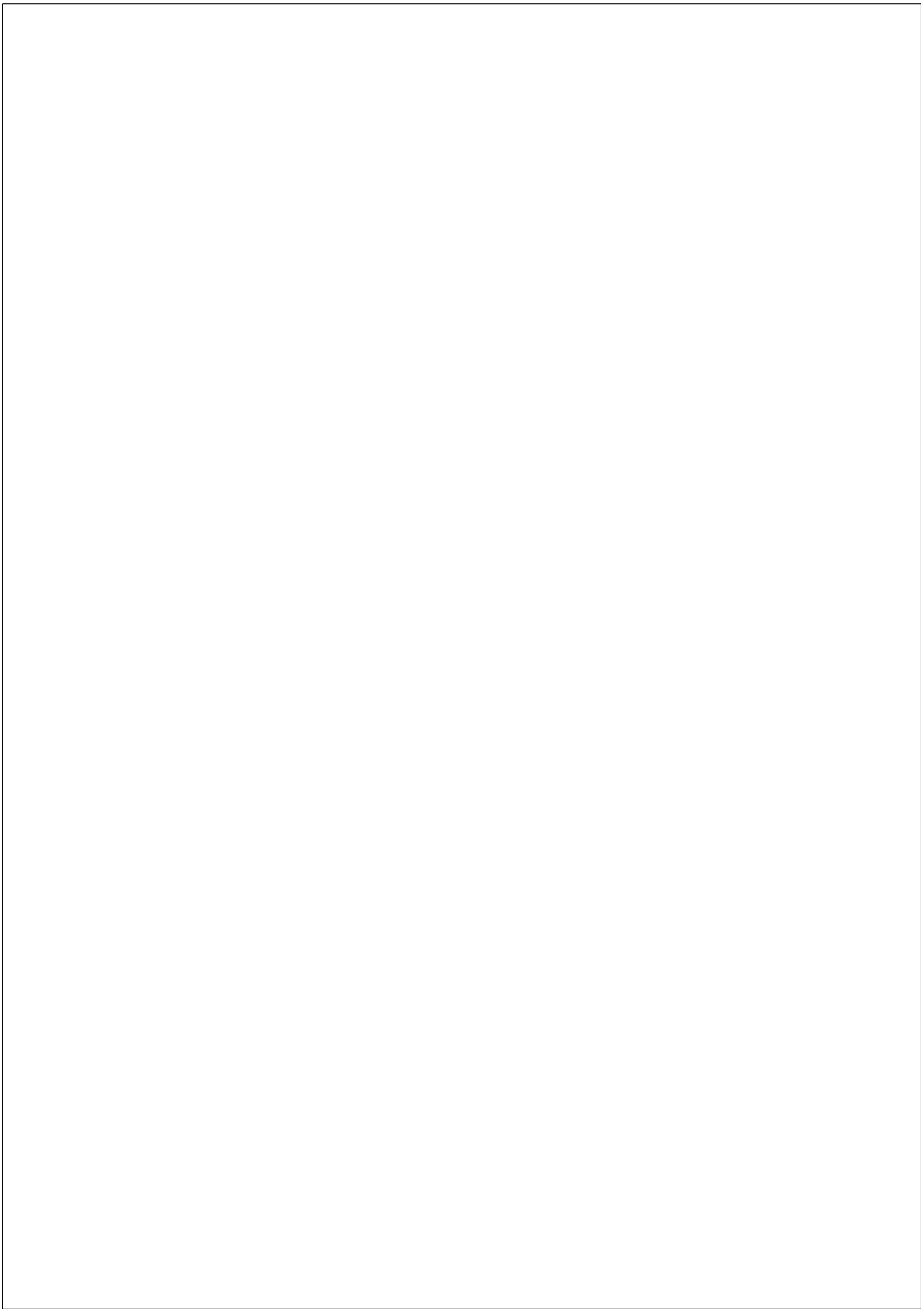
The other dataset is OpenBMAT, a public dataset containing around 27 hours of audio with annotations about the presence of music and its

[3] `https://github.com/DCASE-REPO/dcase_util`

relative loudness with respect to simultaneous non-music sounds. The dataset includes 1647 one-minute audio files, which have been cross-annotated obtaining high partial and full agreement levels for the relative music loudness estimation and the music detection tasks. According to these annotations, more than 50% of the audio files include class changes for the task of music detection, and around 66% include class changes for the task of relative music loudness estimation. We highlight that 35% of OpenBMAT's content is background music at different volumes and that it includes a large variety of non-music sounds resulting from the diversity of broadcast contexts that we have incorporated in the dataset. With these characteristics, we cover all the shortcomings that we mentioned in Section 2.2.1.2, while providing public data, which enables future evaluations to be more transparent, reproducible and comparable. The dataset has already been downloaded 60 unique times[4] from Zenodo[5], and used by Jia et al. (2020), Gimeno et al. (2020) and ourselves (Meléndez-Catalán et al., 2020).

[4]last check on November 27th, 2020

[5]`https://zenodo.org/record/3381249`



Chapter 5

COMPUTATIONAL APPROACHES

In this chapter, we describe two approaches to the task of relative music loudness estimation that we have produced in this doctoral work. The first is the Deep Music Detector: the stable and production-ready technology that BMAT has been using since 2018 to provide the relative music loudness estimation service to its customers. The second approach is more experimental: it is based on the usage of TCNs, and a novel deep learning architecture that combines a TCN with a CNN front-end. We call this network CNN-TCN, and it is inspired by the usage, in the literature reviewed in Section 2.3.2, of CNNs in concatenation with RNNs. With the CNN-TCN, we leverage the capacity of the TCN to model temporal sequences, while boosting its performance with the CNN front-end. This second approach offers highly promising results that we have documented in our third paper (Meléndez-Catalán et al., 2020), which we presented at the 2020 digital audio effects (DAFx) conference. We have yet to incorporate

this second approach into BMAT, but expect to do so in the near future.

These two computational approaches are the final contributions of this thesis. With the Deep Music Detector, we achieve BMAT’s goal of improving the relative music loudness estimation service. Furthermore, as mentioned in Section 1.3, the Deep Music Detector is the first implementation of deep learning inside BMAT. With the experimental approach, we establish the state-of-the-art of the task, and given that we use OpenBMAT to evaluate this approach, we are also providing a benchmark against which future algorithms can be compared in a transparent and reproducible way.

To start the chapter, we devote Section 5.1 to the description of the Deep Music Detector. In this section, we also provide an overview of how this algorithm is used and integrated into BMAT, and discuss its results in two public evaluations. In Section 5.2, we thoroughly describe the TCN and CNN-TCN models, the experiments that we conduct to evaluate their performance, and the results of this evaluation. Finally, in Section 5.3, we provide the conclusions of the chapter.

5.1 Stable approach: the Deep Music Detector

The Deep Music Detector continuously monitors more than 4300 radio stations and TV channels, segmenting the broadcast audio into time intervals of the *Foreground Music*, *Background Music*, and *No Music* classes. In this section, we present its architecture and the process to train it, we discuss several implementation details related to the requirements of the production platform, we provide details about our evaluation methodology, and we show the results of two public evaluations of the algorithm.

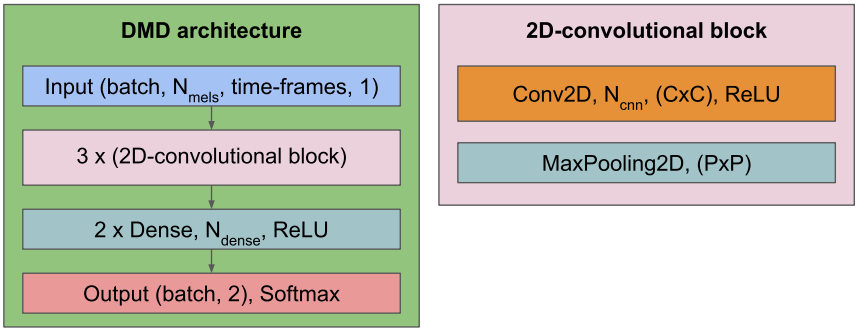


Figure 5.1: Architecture of the Deep Music Detector.

5.1.1 Architecture

In Fig. 5.1, we show the layers that compose the architecture of the Deep Music Detector. It consists of a CNN with three 2D-convolutional blocks followed by two dense layers and the output layer. Each of the 2D-convolutional blocks is composed of a 2D-convolutional layer with N_{cnn} rectified linear units (ReLU) and filter size $C \times C$, and a max-pooling layer. ReLUs are neurons that use a rectifier activation function σ_{relu}, which only lets through the non-negative values as shown in eq. 5.1 for an input $x \in \mathbb{R}$. The stride and dilation rate of the filters of the 2D-convolutional layers is 1 in both axis. The stride of the max-pooling layers is equal to its pooling size $P \times P$.

$$\sigma_{relu}(x) = max(0, x) \tag{5.1}$$

The output layer is a two-neuron dense layer with a softmax activation function. In our case, this function outputs a vector $\hat{\mathbf{y}} \in \mathbb{R}^2$ containing two positive real values that we interpret as the estimated partial loudness

Version	$\mathbf{N_{cnn}}$	**C**	**P**	$\mathbf{N_{dense}}$	**params**
DMDv1	64/32/16	10/5/3	5/3/2	128	97,650
DMDv2	32/64/128	3/3/7	3/5/2	128	453,634

Table 5.1: Architecture's hyper-parameters of *DMDv1* and *DMDv2*. The slashes separate the values for the different 2D-convolutional blocks.

of the music and non-music content at the input, normalized to sum 1. The normalization makes the output independent of the absolute loudness of the analyzed audio. The algorithm adapts the normalized partial loudness values to a classification through the usage of two thresholds, which mark the boundaries between the three classes. By modifying the value of these thresholds we can increase or decrease the precision and recall of each class and adapt the algorithm to any use case and customer. During prediction, once we obtain the classification output, we merge contiguous outputs of the same class creating time intervals of that class. We then smooth short time intervals by modifying their class based on their class and duration, and the class and duration of the contiguous time intervals.

We have developed two versions of the Deep Music Detector: *DMDv1* and *DMDv2*. Both of them share the same generic architecture, but differ in the number of neurons and the size of the filters of the 2D-convolutional layers, and the pooling size of the max-pooling layers. Table 5.1 presents the hyper-parameters of the two versions and their total number of parameters. The hyper-parameters that we set for *DMDv1* are inspired in the work of Doukhan and Carrive (2017). Through a process of trial an error, we later obtained the set of hyper-parameters of *DMDv2*.

5.1.2 Training process

In this section, we first describe how we generate the input features of the Deep Music Detector's CNN from an audio signal, as well as how we process the annotations of the training dataset to assign a ground truth to these input features. We then detail the hyper-parameters that we use to train this network.

5.1.2.1 Input features and ground truth

The dataset that we use to develop the Deep Music Detector is the private dataset described in Section 4.1. To generate the input features of the Deep Music Detector's CNN, we first transcode the audio in the dataset to 8000 samples per second with 16 bits per sample. From this audio, we compute the power spectrogram with a Hanning window of length of 512 samples (64 ms) and a hop size of 128 samples (16 ms). We then apply a Mel filter bank with $N_{mels} = 128$ filters to obtain the Mel-spectrogram, and change its magnitude scale to a logarithmic scale producing the log-magnitude Mel-spectrogram. We cut the log-magnitude Mel-spectrogram in blocks of 128 time-frames, which is equivalent to 2.032 seconds, making the input to the network a square matrix with a 128x128 shape. As explained in the previous section, the output of the Deep Music Detector's CNN is independent of the absolute loudness of the audio under analysis; thus, to prevent the network from learning cues related to it, we apply min-max normalization to each input, ensuring that their values always range from 0 to 1.

In order to train the CNN, each of the network inputs in the training and validation sets need a ground truth $\mathbf{y} \in \mathbb{R}^2$. However, the annotations in the dataset consist of regions (an object of the *Region* model described

in Section 3.2.2) containing a combination of the four classes of the annotation taxonomy: *Music*, *Speech*, *Sound Effects*, and *Audience*, and their partial loudness values. Eq. 5.2 shows how to obtain the normalized partial loudness of the *Music* class $\overline{pl}_{music}$ given the partial loudness value pl_c of each class c in the annotation taxonomy C. As explained in Section 3.2.2, pl_c is an integer in the range of 1 to 5, except for the classes that are not present in the region, which are assigned a partial loudness value of 0. We calculate $\overline{pl}_{music}$ for every region in the annotations, and obtain their ground truth vector $\mathbf{y_r}$ following eq. 5.3.

$$\overline{pl}_{music} = \frac{pl_{music}}{\sum_{c \in C} pl_c} \tag{5.2}$$

$$\mathbf{y}_r = [\overline{pl}_{music}, 1 - \overline{pl}_{music}] \tag{5.3}$$

Note that the time interval covered by a network input can overlap with several regions. As shown in eq. 5.4, we calculate the ground truth $\mathbf{y}$ of an input as the average of the ground truth $\mathbf{y}_r$ of the R regions it overlaps with, weighted by the proportion of time t_r that each region occupies inside the network input.

$$\mathbf{y} = \sum_{r=0}^{R-1} t_r \mathbf{y}_r \tag{5.4}$$

The ground truth assignation process changes in the case of the test set. For this set, the ground truth is not computed for each network input, but for the regions in the annotations of each audio file included in the

set. In addition to this, note that we need to assess the performance of the complete Deep Music Detector, which includes its CNN and the two thresholds that it uses for classification. This means that we need to map the classes contained in every region, and their partial loudness values, to the classes of the relative music loudness estimation task, and not to the vector $\mathbf{y}_r \in \mathbb{R}^2$. We do so using the mapping conditions presented in Section 4.1.2.

5.1.2.2 Training hyper-parameters

The training set comprises the network inputs corresponding to 80% of the audio files in the dataset, while the validation and test sets evenly share the inputs corresponding to the remaining 20%. We train the Deep Music Detector's CNN with the training set for 100 epochs using the ADAM optimizer with learning rate $lr = 0.001$, and the mean square error loss function. We keep the models that produce the lowest loss for the validation set. We shuffle the training data every epoch and present it to the networks in batches of 128 network inputs. We use keras 2.2.4 and tensorflow-gpu 1.12.

5.1.3 Production-ready product

A company's resource limitations play a crucial role when developing an algorithm. The production environment establishes the limits of the resources that an algorithm can consume when deployed into production. It does not matter how good an algorithm is if you cannot run it on your production platform.

5.1.3.1 Production requirements

Characteristics of the production servers such as the operating system, RAM memory, disk space, the number of available threads, etc. need to be taken into account when developing the algorithm. In the case of BMAT, the process of relative music loudness estimation runs in a distributed way over several small servers that are shared with other processes such as the extraction of audio fingerprints or the detection of commercial blocks. Each process must use a single thread, so that, when run in parallel, the number of threads can easily be controlled. They should also avoid RAM peaks that are costly and may harm other processes.

These servers also limit the number and size of the dependencies used: libraries such as Keras,[1] which are very useful to design and train deep learning architectures, might be too heavy and include too many dependencies to be installed in the production platform. It is also necessary to take into account the volume of audio to be analyzed continuously to determine how fast an algorithm must be as well as the number of time intervals it outputs so that the amount of data input to the database is manageable.

5.1.3.2 Code and performance

The Deep Music Detector is implemented using TensorFlow's C API[2] to make it as computationally fast as possible. Fig. 5.2 shows the steps that the Deep Music Detector takes to analyze an audio recording. Basically, the code in C++ receives the paths to the audio and the model in proto-

[1] `https://keras.io/`

[2] `https://github.com/tensorflow/tensorflow/blob/master/tensorflow/c/c_api.h`

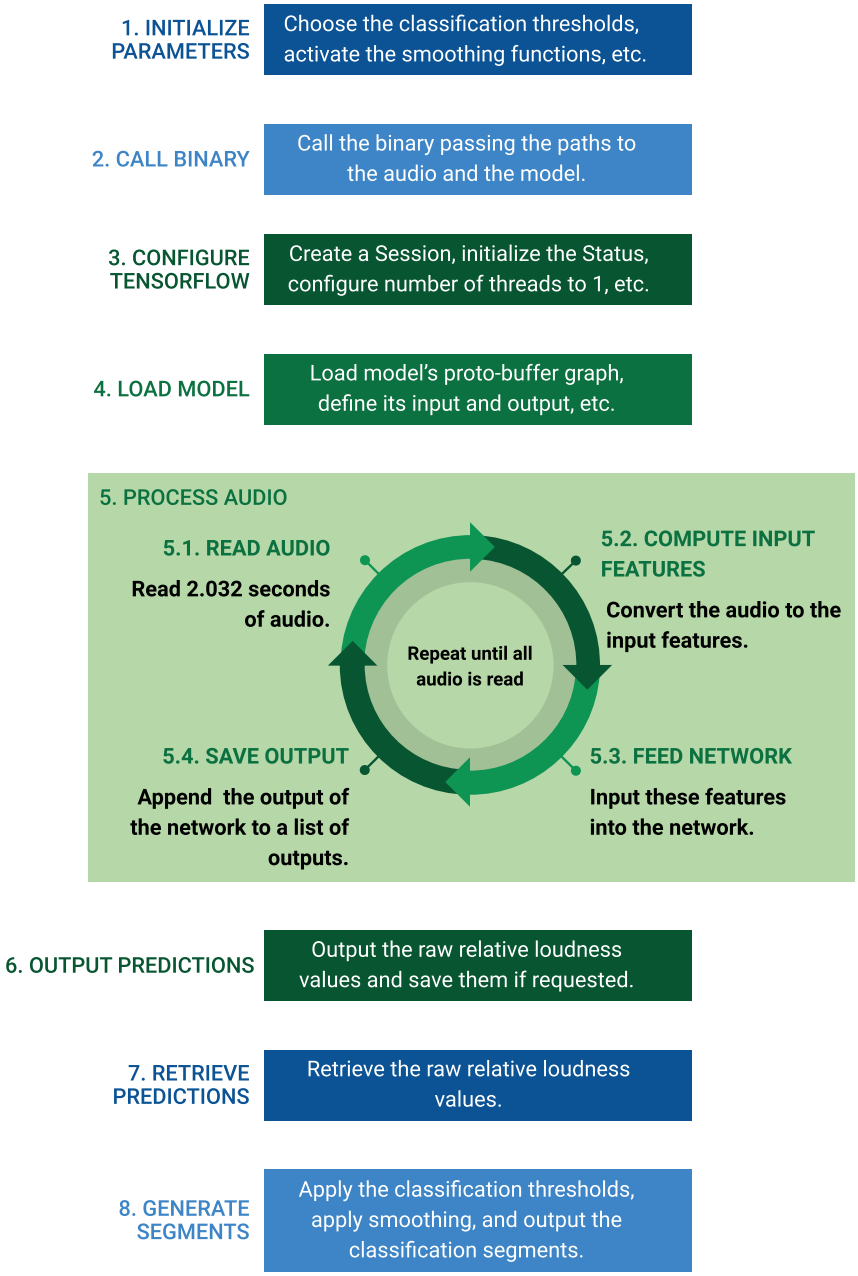


Figure 5.2: Steps of the analysis of an audio file with the Deep Music Detector. In blue, the steps executed by the python wrapper scripts; and in green, the steps executed by the C++ binary.

buffer format, and once the model is loaded, it starts to read the audio block by block. For each block read, it computes the input features and feeds them to the model to get a prediction. The process ends when there is no more audio to read. The algorithm analyzes an hour of audio in approximately 20 seconds using a single thread running on a CPU, and with a RAM memory peak of approximately 54 MB.

We rarely call the C++ code directly as it is wrapped by several python scripts that internally do that as well as setting some parameters, such as the classification thresholds or the activation of the smoothing functions, and retrieving the predictions to turn them into time intervals of the *Foreground music*, *Background music* or *No music* classes. These python scripts are called in a very simple way:

```
1 from python_dmd import DeepMusicDetector
2 dmd = DeepMusicDetector()
3 intervals = dmd.get(audio_path='./path/to/wav')
```

Notice that in this code snippet we are not providing the path to the model. The Deep Music Detector can be installed into a debian operating system as a debian package, which includes the model, the python wrapper and some necessary libraries. The installed python wrapper uses the path to the installed model by default. The repository of the Deep Music Detector also includes a Dockerfile that builds an image of the latest debian LTS release with the Deep Music Detector installed in it.

5.1.4 Evaluation metrics

As mentioned in Section 1.2, BMAT's relative music loudness estimation service has several use cases and types of client. Each customer has their own requirements and each use case has its own particularities, and both can affect how the algorithm must be adapted and evaluated. Specifically, these two variables define what metric is to be maximized by means of tuning algorithm parameters. Often accuracy is not the most important metric, even though it is always crucial to maintain high accuracy levels.

The metrics that we consider for maximization are the balanced accuracy, and the balanced precision and recall of the *No Music* class. There are three out of the four use cases listed in Section 1.2 that require the maximization of metrics other than the accuracy. This includes the most common use case of BMAT's relative music loudness estimation service, which is the calculation of the amount of foreground and background music played in a broadcast channel. As we mentioned in Section 1.1, broadcasters are taxed by the CMOs according to this information; thus, CMOs require the algorithm to reduce as much as possible the number of non-music sounds classified as either *Foreground Music* or *Background Music*. Otherwise, broadcasters might claim that they have been charged unfairly. This means that the algorithm needs to maximize the recall of the *No Music* class. The second use case is to mark parts of the audio that have low probability of containing music to reduce the amount of audio analyzed by BMAT's audio fingerprinting technology. Here we need to maximize the precision of the *No Music* class, so that we are preventing the least amount of music from being identified. The last use case is the analysis of the catalog of others companies in search of non-music content such as audio-books, podcasts, sound effects, etc. In this situation,

we might be interested in maximizing either the precision or the recall of the *No Music* class depending on the goal of the analysis.

5.1.5 Public evaluations

In this section, we first show the results obtained by the Deep Music Detector at the MIREX competitions of 2018 and 2019 for the tasks of music detection and relative music loudness estimation, and compare them with the results of the algorithms of the other participants. We then describe the evaluation of the Deep Music Detector using the OpenBMAT dataset, the results of which we presented in our second publication (Meléndez-Catalán et al., 2019). These results comprise the value of several evaluation metrics, as well as an explanation of the source of the different types of error that the Deep Music Detector makes. We further use this evaluation to demonstrate the potential of having agreement information when assessing an algorithm's performance.

5.1.5.1 MIREX evaluation

We submitted the two versions of the Deep Music Detector described in Section 5.1.1 to the music detection and relative music loudness estimation tasks of the MIREX competition. We call these two versions $DMDv1$ and $DMDv2$. $DMDv2$ preserves the same architecture, input features, and smoothing strategy of the former version, but has approximately 4.6 times more parameters due to the increased number and size of the filters in the 2D-convolutional layers.

In the MIREX competition of 2018, two datasets were used for evaluation: Dataset 1 is a previous version of OpenBMAT that was annotated

with the annotation methodology use for the private dataset, which we described in Section 4.1.2; and Dataset 2 is a dataset containing ten hours of audio from French TV channels and radio stations, provided by the french national institute of audiovisual (INA). It includes archives collected from 1950 up to now.[3] In 2019, there was only one evaluation dataset, which we mentioned in Section 3.4. It contains approximately 50 hours of audio in stereo from TV channels in France, Germany, United States, and Spain, and it was cross-annotated by three annotators using the same annotation methodology as in the OpenBMAT dataset. We only refer to it in this section as it has not been used for any other purpose in relation to this thesis. We call it Dataset 3.

As shown in Table 5.2, in 2018, we submitted *DMDv1* and it obtained the first place in the music detection task for both evaluation datasets out of five participants. Moreover, it was the first algorithm to participate in the relative music loudness estimation task. In 2019, it placed third in both tasks, after *DMDv2*. The first place was claimed by a CNN-TCN prototype that we produced during the elaboration of our third publication (Meléndez-Catalán et al., 2020), which we describe in Section 5.2. In Table 5.3, we show the results of *DMDv1*, *DMDv2*, and the CNN-TCN prototype for the same MIREX events, but for the task of relative music loudness estimation. For the authors that submitted more than one version of the same algorithm, these tables only include the results of the version that performed the best for each dataset.

[3] https://www.music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection#Datasets

Music detection at MIREX 2018					
Algorithm	**Dataset**	**Acc**	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$
Doukhan et al. (2018)	Dataset 1	68.6%	90.5%	38.73%	54.24%
	Dataset 2	92.57%	97.51%	89.5%	93.34%
Jang et al. (2018)	Dataset 1	80.05%	98.24%	59.55%	74.15%
	Dataset 2	94.15%	96.65%	93.15%	94.87%
Choi et al. (2018)	Dataset 1	62.51%	69.15%	39.43%	50.22%
	Dataset 2	79.96%	83.6%	81.37%	82.47%
Marolt (2015)	Dataset 1	68.07%	85.7%	40.26%	54.78%
	Dataset 2	91.5%	97.65%	87.47%	92.28%
DMDv1	Dataset 1	90.49%	91.31%	88.65%	90.96%
	Dataset 2	94.9%	93.99%	98.65%	95.74%
Music detection at MIREX 2019					
Algorithm	**Dataset**	**Acc**	$\mathbf{P_{music}}$	$\mathbf{R_{music}}$	$\mathbf{F_{music}}$
DMDv1	-	87.13%	90.56%	85.13%	87.76%
DMDv2	-	89.28%	91.86%	88.03%	89.9%
CNN-TCN	-	91.78%	90.26%	95.11%	92.62%

Table 5.2: Results of all the algorithms that were submitted to the task of music detection of MIREX 2018 and 2019. In 2018 there were two evaluation datasets for this task, while in 2019 there was only one.

Relative music loudness estimation at MIREX 2018							
Algorithm	**Acc**	$\mathbf{P_{Fg}}$	$\mathbf{R_{Fg}}$	$\mathbf{P_{Bg}}$	$\mathbf{R_{Bg}}$	$\mathbf{P_{No}}$	$\mathbf{R_{No}}$
DMDv1	86.15%	80.25%	77.4%	82.11%	82.1%	90.26%	91.03%
Relative music loudness estimation at MIREX 2019							
Algorithm	**Acc**	$\mathbf{P_{Fg}}$	$\mathbf{R_{Fg}}$	$\mathbf{P_{Bg}}$	$\mathbf{R_{Bg}}$	$\mathbf{P_{No}}$	$\mathbf{R_{No}}$
DMDv1	81.52%	78.74%	73.92%	79.41%	76.47%	83.98%	88.34%
DMDv2	84.14%	87.41%	70.55%	80.35%	81.54%	86.48%	90.79%
CNN-TCN	87.49%	83.75%	82.91%	82.38%	88.96%	93.87%	87.73%

Table 5.3: Results of the algorithms that we submitted to the task of relative music loudness estimation of MIREX 2018 and 2019. In 2018 there were two evaluation datasets, but only Dataset 1 has annotations suitable for this task.

Map	$\mathbf{Acc^b}$ (%)	$\mathbf{P^b_{Fg}}$ (%)	$\mathbf{R^b_{Fg}}$ (%)	$\mathbf{P^b_{Bg}}$ (%)	$\mathbf{R^b_{Bg}}$ (%)	$\mathbf{P^b_{No}}$ (%)	$\mathbf{R^b_{No}}$ (%)
MD	89.15	87.61	91.2	-	-	90.83	87.1
RMLE	82.11	85.1	82.25	76.69	76.99	84.59	87.1

Table 5.4: Performance of *DMDv1* on the OpenBMAT dataset for the tasks of relative music loudness estimation and music detection. In this table, *Fg* stands both for *Foreground Music*, in the case of the relative music loudness estimation task, and *Music*, for the music detection task.

5.1.5.2 Evaluation using OpenBMAT

We carried out this evaluation for our second publication (Meléndez-Catalán et al., 2019). The evaluated algorithm is *DMDv1*, and the testing dataset is the entire OpenBMAT dataset. The results of this evaluation establish a baseline for future relative music loudness estimation and music detection algorithms. Furthermore, we use them for error analysis, as well as to demonstrate the usefulness of having access to information about the annotators agreement in a dataset.

Table 5.4 presents the evaluation results. To compute the values in this table, we have run *DMDv1* over the entire dataset using the ground truth resulting from merging the three individual annotations. We apply the same merging strategy, described at the end of Section 5.1.2.1, that we use for the test set of the private dataset. We only discard the time intervals with no agreement, which only represents 0.21% of the total duration of the dataset in the case of the relative music loudness estimation task. In the case of music detection, there cannot be disagreement. We observe a balanced accuracy lower than 90% for both evaluations, which indicates that there is still room for improvement for future algorithms. Note that

Ground truth	*DMDv1* classified as		
	Fg	**Bg**	**No**
Fg	**82.25%**	11.56%	6.19%
Bg	13.34%	**76.99%**	9.67%
No	1.06%	11.84%	**87.1%**

Table 5.5: Balance-independent relative music loudness estimation confusion matrix for the $DMDv1$.

the evaluation statistics for the music detection task are significantly better than for the relative music loudness estimation task. This happens because the division of the *Music* class into the *Foreground Music* and *Background Music* classes introduces new errors. Using the 3×3 balanced confusion matrix shown in Table 5.5, we can distinguish between different types of errors:

- Errors between *Foreground Music* and *Background Music*: this type of error is often related to a certain grey zone between the two classes. As we have seen in Fig. 4.6, this is also where annotators have the least agreement, and thus, it can be a challenging content to classify.

- *Foreground Music* classified as *No Music*: these errors are typically related to certain music genres. The algorithm tends to make this kind of errors when analyzing music with a prominent voice part, such as a cappella singing, opera, or hip-hop, or music with a lot of percussion.

- *Background Music* classified as *No Music*: these errors are mainly related to the music's volume. They happen due to the incapacity

Ground truth	*DMDv1* classified as	
	Music	**No Music**
Music	**91.2%**	8.8%
No Music	12.9%	**87.1%**

Table 5.6: Balance-independent music detection confusion matrix for the *DMDv1*.

of the algorithm to detect music with a very low loudness in comparison to simultaneous non-music sounds.

- *No Music* classified as *Background Music*: these errors are commonly related to a certain ambiguity between what is music and what is not. This type of error includes, mainly, non-music sounds with certain musical features, such as a recognizable pitch or rhythm. For example, sounds like a church bell, a ringing phone or background noises with one or more prominent frequencies.

- *No Music* classified as *Foreground Music*: the source of this type of errors is mainly the imprecision of the algorithm or the annotation in the positioning of boundaries between time intervals of these two classes.

Table 5.6 shows the result of mapping the 3×3 confusion matrix to the classes of the music detection task. The former errors between *Foreground Music* and *Background Music* disappear, which leads to a high balanced *Music* recall.

The concept of agreement between annotators is closely related to the existence or not of an actual ground truth. If all annotators coincide in the annotation of a time interval, this time interval probably has a clear ground truth according to the given taxonomy. If the content of the time

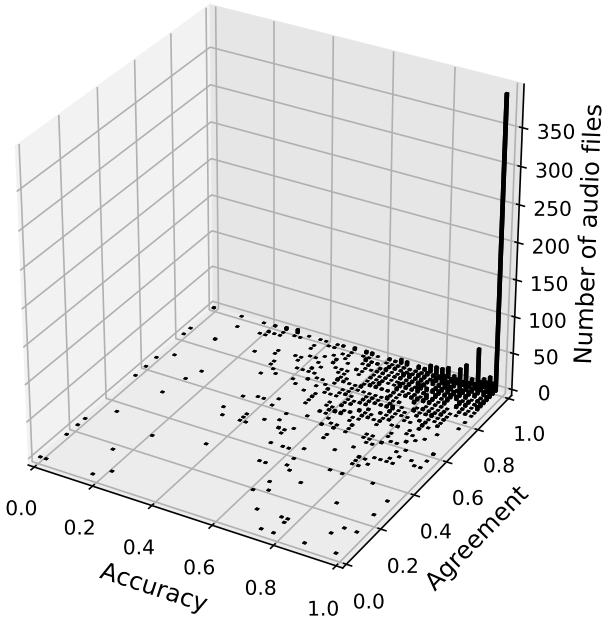


Figure 5.3: Audio file distribution by full agreement applying the RMLE mapping and the accuracy achieved by $DMDv1$ when evaluated against the ground truth resulting from merging the three individual annotations.

interval is too ambiguous for three humans to agree, then it might be that an actual ground truth does not exist, and thus, this time interval might not be suitable to train an algorithm or evaluate its performance. It is in the hands of potential users of OpenBMAT to decide if they want to train or test their algorithms using only the content with full agreement or, alternatively, include time intervals with partial or no agreement. Note that these ambiguous time intervals might generate a glass-ceiling effect for the algorithm’s accuracy when used for testing.

Fig. 5.3 shows the distribution of OpenBMAT’s audio files by their

$\%FA_{af}$, in the horizontal axis, and the accuracy that $DMDv1$ obtained for that audio file using the ground truth resulting from merging the three individual annotations, in the vertical axis. Both agreement and accuracy values are computed using the relative music loudness estimation classes. By adding the agreement axis, we can distinguish between errors in audio files with a clear ground truth, and errors in audio files with an ambiguous ground truth. Potential users of this dataset will probably find it more beneficial to focus on the first type of errors in order to improve their algorithms. Using the intervals in the agreement metadata (see Section 4.2.4), the users can know if the errors correspond to a time interval with full, partial or no agreement. Therefore, having agreement information can help us better judge the severity of an algorithm's error, and have a better insight during its evaluation. Note that in Fig. 5.3 there are more audio files with low accuracy and high agreement than audio files with low accuracy and low agreement. This seems contradictory, but we need to take into account that audio files with low agreement are scarce in OpenBMAT.

5.2 Experimental approach: the TCN and CNN-TCN architectures

With the Deep Music Detector, we already met BMAT's goal of developing a better relative music loudness estimation algorithm. However, despite its remarkable performance, the Deep Music Detector is admittedly a fairly simple and shallow CNN with no particular feature that makes it especially suitable for the task of relative music loudness estimation. In our last publication (Meléndez-Catalán et al., 2020), we study the use-

fulness of TCNs for this task. As we have already explained, TCNs are a type of CNN with the ability to model temporal sequences, which we consider a fundamental characteristic when analyzing temporally correlated signals such as music. Moreover, inspired by the usage of CNNs to boost the performance of RNNs, we attach a CNN front-end to the TCN producing a novel type of network that we call CNN-TCN and that offers very promising results. We presented this study at the 2020 DAFx conference.

In this section, we first detail the networks that we propose for the task of relative music loudness estimation: the TCN and the CNN-TCN. We include the particularities of their architecture, an explanation of the input features and the ground truth that they use, and the strategy we apply to smooth their classification output. After this, we describe the experiment that we carried out to evaluate their performance, and present the results of this evaluation.

5.2.1 Architectures

The TCN model is formed by a stack of six residual blocks. In the right-bottom part of Fig. 5.4, we show the structure of the residual blocks that we use in this paper. Our residual blocks contain two 1D-convolutional layers as proposed by Bai et al. (2018). All 1D-convolutional layers have N_{tcn} neurons with non-causal filters (Lemaire and Holzapfel, 2019) of length L and a 1D-spatial dropout rate dr. However, only the first 1D-convolutional layer uses a rectifier activation function: we have removed the activation function of the second 1D-convolutional layer so that the modifications learned by the residual block (see Section 2.1.1.3) may include negative values as originally designed by He et al. (2016). The

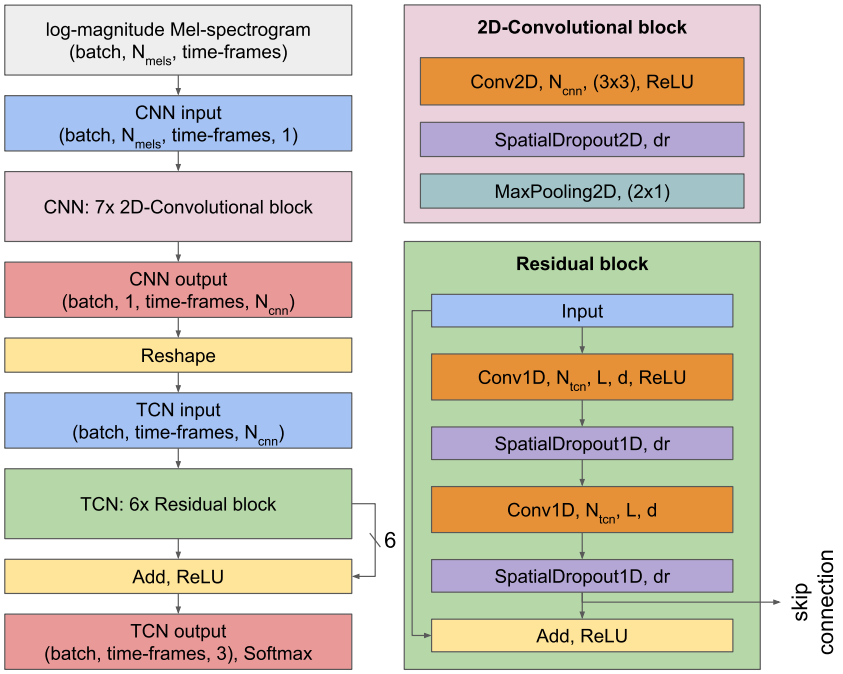


Figure 5.4: (Left) complete CNN-TCN architecture. (Right-top) structure of the convolutional block. (Right-bottom) structure of the residual block.

1D-convolutional layers of each subsequent residual block have a higher dilation rate d starting at 1 and increasing by a factor of 2 for each block. As we explained in Section 2.1.1.3, the first 1D-convolutional layer of the TCN reads the log-magnitude Mel-spectrogram as N_{mels} scalar temporal sequences by interpreting the frequency axis as channels.

In the left part of Fig. 5.4, we show the CNN-TCN model, which is a combination of a CNN front-end and the TCN described above. The CNN front-end consists of a stack of seven blocks that comprise a 2D-convolutional layer with N_{cnn} ReLUs containing 3×3 filters and 2D-

spatial dropout rate dr, and a max-pooling layer. We apply the max-pooling only to the frequency axis reducing its dimensionality by a factor of 2 for each block until it is equal to 1. This time, the TCN reads the output of the CNN as N_{cnn} scalar temporal sequences and models their evolution.

Concatenating a CNN and a TCN does not necessarily produce a model with more parameters than the TCN model alone. Eq. 5.5 shows how the number of parameters (P) in a 1D-convolutional layer of the TCN depends on the number of input channels N_{ch}, the number of filters N_{tcn}, their length L, and the length B of the bias vector. Adding the CNN front-end transforms the number of channels that we input to the TCN from $N_{ch} = N_{mels}$ to $N_{ch} = N_{cnn}$. If we reduce the number of channels by using a low N_{cnn} value, we can significantly decrease the number of parameters of the first 1D-convolutional layer of the TCN compensating the number of parameters added by the CNN itself.

$$P = N_{ch}N_{tcn}L + B \tag{5.5}$$

The output layer for both the TCN and CNN-TCN architectures has three neurons, each of which corresponds to one of the classes of the relative music loudness estimation task. Using a softmax activation function, the networks outputs a vector $\hat{\mathbf{y}} \in \mathbb{R}^3$ for each time-frame, containing the estimated probability of each class. This means that they offer predictions at a frame-level regardless of the input size in the temporal dimension. We assign the class with the highest probability to each time-frame. We refer to a set of contiguous time-frames of the same class as a time interval of that class. The time interval starts at the beginning of the first time-frame and ends at the end of the last time-frame, in chronological order.

5.2.2 Input features

To generate the input features of the TCN and CNN-TCN architectures, we use audio transcoded to 8000 samples per second with 16 bits per sample. From the audio data, we compute the power spectrogram with a Hanning window with a length 512 samples (64 ms) and a hop size of 128 samples (16 ms). We then apply a Mel filter bank with $N_{mels} = 128$ filters to obtain the Mel-spectrogram, and change its magnitude scale to a logarithmic scale producing the log-magnitude Mel-spectrogram. We cut the log-magnitude Mel-spectrogram in blocks of 625 time-frames, which is equivalent to 10 seconds, making the input to the network a matrix with a 128x625 shape. The classes of the relative music loudness estimation task are independent of the absolute loudness of the audio under analysis; thus, to prevent our TCN and CNN-TCN models from learning cues related to it, we apply min-max normalization to each input, ensuring that their values always range from 0 to 1. In Section 5.2.4.3, we explain the process to generate the ground truth of these network inputs.

5.2.3 Smoothing

Classifying at a frame-level allows an algorithm to be very precise in detecting a change of class; however, it also makes it prone to produce short erroneous time intervals that make the classification noisy. To solve this issue, we apply a smoothing strategy to the output of our models: we use a sliding window that assigns the most represented class across all time-frames covered by the window to its central time-frame.

5.2.4 Experimental setup

In this experiment, we carry out two grid searches over a total of four hyper-parameters using the OpenBMAT dataset to find the configuration that produces the best possible TCN (TCN^{best}) and CNN-TCN ($CNN\text{-}TCN^{best}$) models. We compare these models with an adaptation of the two Deep Music Detector versions ($DMDv1$ and $DMDv2$) that we introduced in Section 5.1.5.1. In what follows, we group them as DMD-based algorithms. This experiment also includes the training of the DMD-based algorithms. We impose two restrictions when searching for TCN^{best} and $CNN\text{-}TCN^{best}$:

- TCN^{best} must have a lower number of parameters than the DMD-based algorithms.
- $CNN\text{-}TCN^{best}$ must have a lower number of parameters than TCN^{best}.

With these restrictions we make sure that any improvement comes from a more appropriate architecture and not just from an increase in the networks learning capacity. In the rest of this section, we describe the DMD-based algorithms, we detail the training process, we explain how we use the OpenBMAT dataset for training and testing, and we present the evaluation metrics that we use in the experiment.

5.2.4.1 Baselines: DMD-based algorithms

As mentioned before, the Deep Music Detector is a regression algorithm; unfortunately, the annotations in OpenBMAT are designed for classification and not for regression. For this reason, in this experiment, we need

to adapt $DMDv1$ and $DMDv2$ for classification. To do so, we only need to replace the two-neuron output layer with a three-neuron output layer. In this way, we can train these algorithms to output the probability of each of the three classes of the relative music loudness estimation task. We call the adapted algorithms $DMDv1_{clf}$ and $DMDv2_{clf}$. In all other respects, the DMD-based algorithms are identical to their corresponding Deep Music Detector versions. We even apply the same rules to smooth their predictions: we modify the class of a particular time interval based on its class and duration, and the class and duration of the contiguous time intervals.

Despite $DMDv2_{clf}$ having approximately 4.6 times more parameters than $DMDv1_{clf}$, the difference in accuracy between them in the MIREX 2019 competition was approximately of 2 percentage points for the task of music detection and 3 percentage points for the task of relative music loudness estimation. We find it difficult to judge what architecture is more appropriate for these two tasks, which is why we include both of them in the experiment. After the adaptation, $DMDv1_{clf}$ and $DMDv2_{clf}$ have a total of 97,779 and 453,763 parameters, respectively.

We train the DMD-based algorithms for 100 epochs using the ADAM optimizer with learning rate $lr = 0.001$, and the categorical cross-entropy loss function. We weight the loss function to compensate for the imbalance in terms of the number network inputs per class of the training and validation sets. We keep the models that produce the lowest loss for the validation set. We shuffle the training data every epoch and present it to the networks in batches of 128 network inputs. We use keras 2.2.4 and tensorflow-gpu 1.12. We apply a range of 2D-spatial dropout rates to the 2D-convolutional layers to avoid overfitting.

5.2.4.2 TCN and CNN-TCN models

The first models that we train are the TCN models. We do so through a grid search over the hyper-parameters described in Section 5.2.1: the number of filters N_{tcn}, the filter length L, and the dropout rate dr of the 1D-convolutional layers. L allows us to modify the receptive field of the TCN without affecting the model's architecture. With N_{tcn} and dr we experiment with the learning capacity of the network and its regularization, respectively. We train 40 models using the following set of values for each hyper-parameter:

- $N_{tcn} \in [16, 32]$
- $L \in [3, 5, 7, 9, 11]$
- $dr \in [0.0, 0.05, 0.1, 0.15]$

We then combine these TCN models with two CNNs to generate 80 CNN-TCN models. The 2D-convolutional layers of these CNNs have N_{cnn} 3×3 filters where:

- $N_{cnn} \in [16, 32]$

We choose sets of hyper-parameter values that produce a majority of networks with a number of parameters lower than the number of parameters of $DMDv1_{clf}$. Given that these networks require regularization through the usage of dropout layers, we considered that they have sufficient capacity to absorb the training dataset.

We train the TCN and CNN-TCN models for 100 epochs using the ADAM optimizer with learning rate $lr = 0.001$, and the categorical cross-entropy loss function. We weight the loss function to compensate for the

imbalance in terms of the number of time-frames per class of the training and validation sets. We keep the models that produce the lowest loss for the validation set. We shuffle the training data every epoch and present it to the networks in batches of 128 network inputs. We use keras 2.2.4 and tensorflow-gpu 1.12.

5.2.4.3 Dataset

We use OpenBMAT, the public dataset described in Section 4.2, as the training and evaluation dataset for this experiment. Using a public dataset is necessary for the experiment to be transparent and reproducible, and for its results to be comparable. In Section 4.2.3.1, we mentioned that OpenBMAT comes with ten predefined splits containing approximately 15% *Foreground Music*, 35% *Background Music* and 50% *No Music* each. During training, we use nine of them: the first eight for the training set and the ninth for the validation set. The tenth split constitutes the testing set. From each split we only use the audio excerpts that have at least partial agreement, i.e., the parts where at least two annotators agree. This supposes 99.79% of the content for the relative music loudness estimation classes.

In Section 5.2.1, we explained that the TCN and the CNN-TCN architectures output a vector $\hat{\mathbf{y}} \in \mathbb{R}^3$ for each time-frame, containing the estimated probabilities of each of the relative music loudness estimation classes. Thus, in order to train these networks, we need to transform the annotations of OpenBMAT corresponding to the training and validation sets to a set of frame-level one-hot ground truth vectors with length 3. Fig. 5.5 illustrates this transformation. We first apply the RMLE mapping and merge the annotations of the three annotators keeping the class that pro-

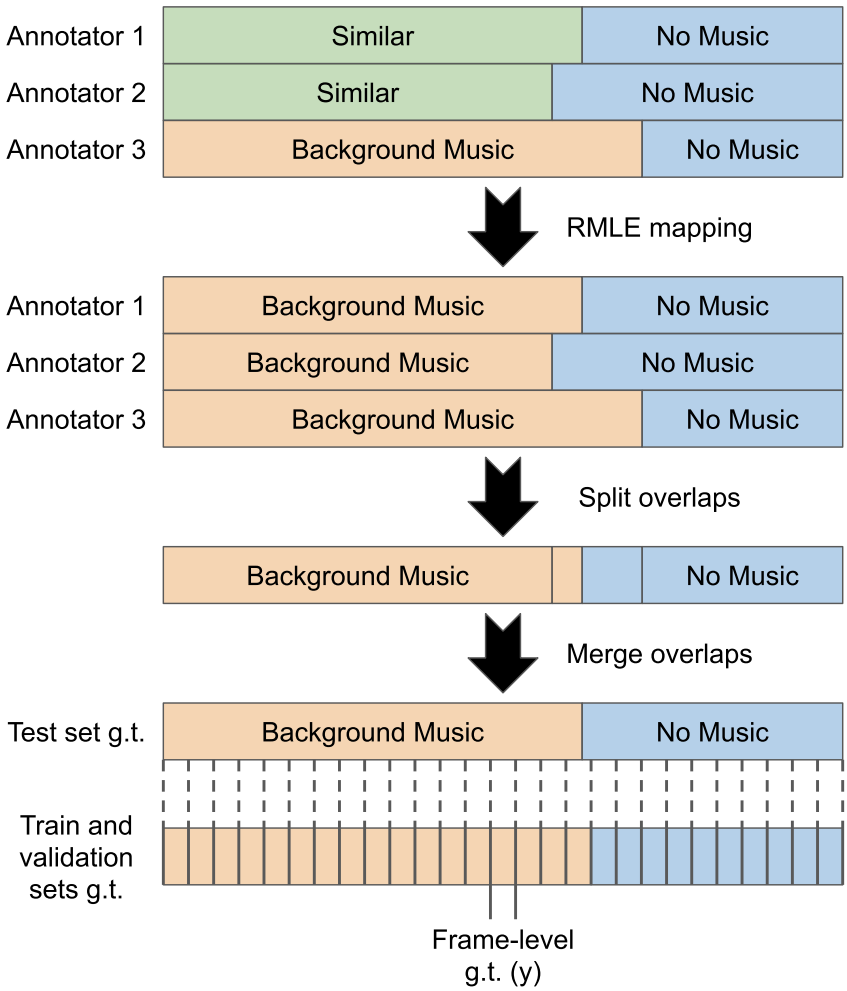


Figure 5.5: Ground truth generation process for the training, validation and test sets of the OpenBMAT dataset.

duces partial or full agreement at each point in time. Then, we assign to each time-frame the class that is majority inside the time interval covered by the time-frame. In the case of the DMD-based algorithms, the process is the same, but we assign a single class to the entire network input, and the assigned class is the one that is majority inside the time interval covered by the entire network input. In order to produces a higher number of network inputs, we introduce an overlap between contiguous inputs of 50%. This results in 12,892 inputs for training and 1,616 inputs for validation in the case of the TCN and CNN-TCN models, and 74,585 inputs for training and 9,286 inputs for validation in the case of the DMD-based algorithms.

For the test set, the process in Fig. 5.5 stops at the penultimate step: the ground truth is computed for the regions (objects of the *Region* model described in Section 3.2.2) in the annotations of each audio file included in the test set, and not for the network inputs. This means that the resulting ground truth is the same for the DMD-based algorithms, and the TCN and CNN-TCN models, which is mandatory to be able to compare them.

5.2.4.4 Metrics

To evaluate a model, we rely on the following metrics: the balanced accuracy (Acc^b), and the balanced precision (P_c^b) and recall (R_c^b) of each of the classes of the relative music loudness estimation task. We also add a new event-based metric that we call ratio of intervals (RI) and define as the ratio between the number of predicted time intervals I_{pr}, and the average number of ground truth time intervals for all annotators I_{gt} across N audio files. We formally define RI in Eq. 5.6. This metric is limited to the evaluation of the number of output events: it provides relevant information

about how noisy a model is regardless of how correct its predictions are, which is another characteristic of its performance that complements the insights that we can extract from the value of the segment-based metrics. The optimal value of RI is 1.

$$RI = \frac{\sum_{n=0}^{N} I_{pr_n}}{\sum_{n=0}^{N} I_{gt_n}} \tag{5.6}$$

5.2.5 Results and discussion

As shown in Table 5.7, both $CNN\text{-}TCN^{best}$ and TCN^{best} models outperform the DMD-based algorithms in terms of Acc^b despite using fewer parameters. TCN^{best} uses the following hyper-parameters:

- $N_{tcn} = 32$
- $L = 5$
- $dr = 0.15$

The receptive field of this model is 253 time-frames, which is equivalent to approximately 4 seconds. After the smoothing, TCN^{best} obtains a Acc^b value 4.6 percentage points higher than $DMDv1_{clf}$ using 12.4% fewer parameters and 3.2 percentage points higher than $DMDv2_{clf}$ using 81% fewer parameters. $CNN\text{-}TCN^{best}$ uses the following hyper-parameters:

- $N_{cnn} = 32$
- $N_{tcn} = 16$
- $L = 7$

- $dr = 0.15$

The receptive field of this model is 379 time-frames, which is equivalent to approximately 6.1 seconds. There is an improvement in Acc^b, after the smoothing, of 8.4 percentage points with respect to $DMDv1_{clf}$ using 17% fewer parameters and of 7 percentage points with respect to $DMDv2_{clf}$ using 82% fewer parameters. The improvement with respect to TCN^{best} is of 3.8 percentage points while using 5.5% less parameters. Obtaining better results with fewer parameters implies that the architecture makes a more efficient usage of its parameters, and thus, is more appropriate for the task. Note that TCN^{best} and $CNN\text{-}TCN^{best}$ consider, respectively, twice and thrice more temporal context to classify than the DMD-based algorithms.

As shown in Table 5.8 and Table 5.9, $CNN\text{-}TCN^{best}$ provides a significant improvement with respect to $DMDv1_{clf}$ and $DMDv2_{clf}$ in the detection of background music, which is one of the most challenging types of content due to the low volume of the music (Meléndez-Catalán et al., 2019). $CNN\text{-}TCN^{best}$ correctly classifies 34.9% of the background music that $DMDv2_{clf}$ cannot detect and misclassifies as *No Music*. This percentage rises to 39.2% in the case of $DMDv1_{clf}$. However, the statistics for the *Background Music* class show that there is still room for improvement for the relative music loudness estimation and music detection tasks.

In the left part of Fig. 5.6, we observe that all CNN-TCN models achieve better Acc^b than any TCN model. This figure includes all TCN and CNN-TCN models. Table 5.10 shows a comparison between TCN^{best} and a CNN-TCN model that shares the same hyper-parameters. Note that the CNN-TCN model has fewer parameters, but still outper-

Model	$\mathbf{Acc^b}$	$\mathbf{P^b_{Fg}}$	$\mathbf{R^b_{Fg}}$	$\mathbf{P^b_{Bg}}$	$\mathbf{R^b_{Bg}}$	$\mathbf{P^b_{No}}$	$\mathbf{R^b_{No}}$	**RI**	**params**
$DMDv1_{clf}$	81.36%	86.98%	84.22%	73.12%	75.9%	84.49%	83.97%	1.76	97,779
$DMDv1_{clf}$ (S)	81.7%	88.35%	79.57%	71.98%	79.79%	86.54%	85.74%	0.83	97,779
$DMDv2_{clf}$	82.57%	85.79%	86.88%	75.42%	75.47%	86.51%	85.34%	1.62	453,763
$DMDv2_{clf}$ (S)	83.04%	86.56%	83.92%	74.83%	77.86%	88.21%	87.33%	0.81	453,763
TCN^{best}	85.76%	90.08%	89.21%	79.52%	80.39%	87.79%	87.69%	39.27	85,635
TCN^{best} (S)	86.27%	90.68%	89.14%	79.85%	81.63%	88.52%	88.05%	1.54	85,635
$CNN\text{-}TCN^{best}$	90.05%	91.43%	**93.38%**	**86.18%**	84.58%	92.43%	92.18%	11.41	80,963
$CNN\text{-}TCN^{best}$ (S)	**90.06%**	**91.81%**	92.79%	85.73%	**85.2%**	**92.61%**	**92.19%**	**1.14**	80,963

Table 5.7: Statistics of $DMDv1_{clf}$, $DMDv2_{clf}$, TCN^{best} and $CNN\text{-}TCN^{best}$ with and without smoothing (S).

Ground Truth	$DMDv1_{clf}$ Classified As			$DMDv2_{clf}$ Classified As		
	Fg	**Bg**	**No**	**Fg**	**Bg**	**No**
Fg	**79.54%**	18.65%	1.77%	**83.92%**	15.22%	0.86%
Bg	8.64%	**79.79%**	11.56%	11.33%	**77.86%**	10.8%
No	1.85%	12.41%	**85.74%**	1.7%	10.97%	**87.33%**

Table 5.8: Balance-independent confusion matrices for the $DMDv1_{clf}$ and $DMDv2_{clf}$ algorithms with smoothing.

Ground Truth	TCN^{best} Classified As			$CNN\text{-}TCN^{best}$ Classified As		
	Fg	**Bg**	**No**	**Fg**	**Bg**	**No**
Fg	**89.14%**	10.16%	0.7%	**92.79%**	6.89%	0.32%
Bg	7.66%	**81.63%**	10.72%	7.76%	**85.2%**	7.03%
No	1.51%	10.44%	**88.05%**	0.51%	7.29%	**92.19%**

Table 5.9: Balance-independent confusion matrices for the TCN^{best} and $CNN\text{-}TCN^{best}$ models with smoothing.

forms TCN^{best} in terms of Acc^b. This further proves that using a CNN front-end improves performance.

The Ratio of Intervals in Table 5.7 shows that both TCN^{best} and $CNN\text{-}TCN^{best}$ predict a number of time intervals that is much higher than the number of time intervals in the ground truth. In this particular aspect, $DMDv1_{clf}$ and $DMDv2_{clf}$ are superior to our models, which are prone to generate noise in the form of short erroneous time intervals, especially near class changes. The bottom part of Fig. 5.7 shows an example of this noise around second 8. To remove this noise, we apply the smoothing strategy described in Section 5.2.3. Using the validation set, we evaluate the impact on RI and Acc^b of six window sizes ranging from 0.5 to 3 seconds with steps of 0.5 seconds. We find an optimal window size of 2 seconds for both models. This window size produces a strong

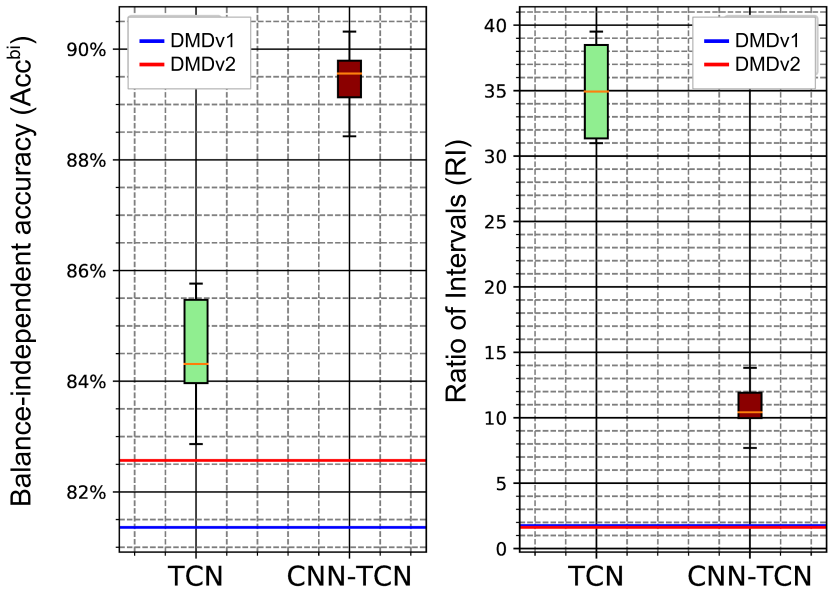


Figure 5.6: Comparison between $DMDv1$, $DMDv2$ and all the TCN and CNN-TCN models in terms of Acc^b and RI without smoothing. The horizontal lines at the bottom correspond to $DMDv1$ and $DMDv2$.

decrease in RI and a light improvement of Acc^b. Smaller window sizes do not decrease RI enough neither increase Acc^b significantly. Larger window sizes start decreasing Acc^b as they can remove correct time intervals of 1 second or larger. As shown in Table 5.7, after the smoothing, TCN^{best} and $CNN\text{-}TCN^{best}$ predict 54% and 14% more time intervals with respect to the time intervals in the ground truth, respectively. Note, in the right part of Fig. 5.6, that all CNN-TCN models produce significantly less noise than any TCN model. This indicates that the CNN front-end also helps in reducing this phenomenon.

Arch	$\mathbf{Acc^b}$	$\mathbf{N_{cnn}}$	$\mathbf{N_{tcn}}$	L	params
TCN^{best}	85.76%	-	32	5	85,635
$CNN\text{-}TCN$	**89.15%**	16	32	5	**78,211**

Table 5.10: Comparison between TCN^{best} and a CNN-TCN model with the same hyper-parameters except for the dropout rate (dr). We pick the best dropout rate for each model. We do not apply any smoothing.

Algorithm	Computation time (s/h)
$DMDv1_{clf}$	41.99 (0.16)
$DMDv2_{clf}$	19.65 (0.56)
TCN^{best}	**8.43** (0.46)
$CNN\text{-}TCN^{best}$	70.1 (3.52)

Table 5.11: Mean and standard deviation (between parenthesis) of the computation time in seconds per analyzed hour of audio of each algorithm over 10 runs using one thread and a CPU.

Another relevant characteristic of an algorithm, especially in the industrial context, is the computation time. In Table 5.11, we show the computation time of each model. We observe that the TCN^{best} is the fastest architecture with less than 9 seconds per hour of audio analyzed, while the $CNN\text{-}TCN^{best}$ is the slowest one with approximately 70 seconds per hour of audio analyzed.

Analyzing the duration of the $CNN\text{-}TCN^{best}$ errors when we apply no smoothing shows that almost 90% of the misclassified time intervals have a duration that is less than or equal to 0.2 seconds. These errors amount to approximately 16% of the misclassified time and come mainly from a noisy classification and precision errors in class changes during the annotation or the classification. Listening to the errors with durations that are equal to or greater than 3 seconds, which represent approximately 50%

of the misclassified time, we discover several patterns. $CNN\text{-}TCN^{best}$ misclassifies:

- Loud and mixed non-music sound effects as in action films (*No Music*) as *Background Music*.
- Speech mixed with background non-music noises with an identifiable pitch such as engine sounds (*No Music*) as *Background Music*
- Low volume background music (*Background Music*) classified as *No Music*. Especially percussion music, live music and tones.
- Loud live music mixed with applauses, cheering and other audience sounds (*Foreground Music*) as *Background Music*.

We have analyzed the features extracted by the CNN front-end. The top and mid parts of Fig. 5.7 show how the CNN front-end works as a feature extractor transforming and reducing the dimensionality of the input log-magnitude Mel-sectrogram. In the example, we observe four consistent patterns in the generated features corresponding to four sound combinations: (4) isolated music, (1) isolated speech, (3) mixed music and speech, (2) silence. The bottom part of Fig. 5.7 presents the $CNN\text{-}TCN^{best}$ classification and the ground truth for the input at the top of the figure. We observe an annotation precision error around second 3, which is shorter than 0.2 seconds and an example of noisy classification in the *Foreground Music* time interval between seconds 8 and 9.

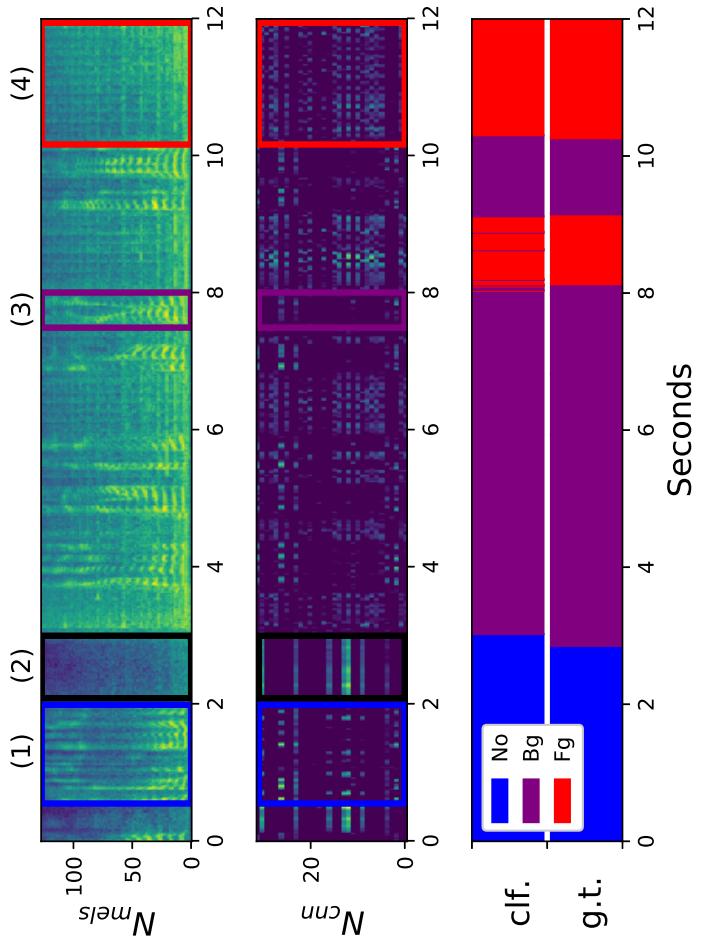


Figure 5.7: (top) Example of the log-magnitude Mel-spectrogram, which we use as input features for both the TCN and CNN-TCN architectures. (mid) Output of the CNN in the CNN-TCN architecture for these features. (bottom-top) $CNN\text{-}TCN^{best}$ classification for these features without smoothing. (bottom-bottom) Ground truth of these features.

5.3 Conclusions

In this chapter, we have presented two approaches to the task of relative music loudness estimation. The first approach is the Deep Music Detector: the stable and production-ready technology that BMAT has been using since 2018 for its relative music loudness estimation service. Its optimized implementation allows it to process one hour of audio in approximately 20 seconds. We have provided an overview of its architecture, input features, and training dataset, as well as a description of how we have integrated it into BMAT and the steps involved in its analysis of audio recordings. In addition, we have also reported on its performance in the MIREX competition, where it obtained outstanding results, and in the evaluation that we carried out for our second published paper (Meléndez-Catalán et al., 2019).

The second approach includes two architectures for the task of relative music loudness estimation: the TCN, and the CNN-TCN. The CNN-TCN is a novel architecture that consists of the combination of a TCN with a CNN front-end. Using the OpenBMAT dataset for training and testing, we have compared these architectures against the DMD-based algorithms: $DMDv1_{clf}$ and $DMDv2_{clf}$. We have named TCN^{best} the TCN model that performs the best while using fewer parameters than $DMDv1_{clf}$ and $DMDv2_{clf}$, and we have named $CNN\text{-}TCN^{best}$ the CNN-TCN model that performs the best while using fewer parameters than TCN^{best}. Producing better results with fewer parameters means that our architectures make a more efficient usage of its parameters, and thus, are more appropriate for the task.

The results of the evaluation after the smoothing show that, in terms of Acc^b, TCN^{best} outperforms both $DMDv1_{clf}$ and $DMDv2_{clf}$, and

$CNN\text{-}TCN^{best}$ outperforms TCN^{best}. Additionally, our models provide a better classification of background music. We observe that both TCN^{best} and $CNN\text{-}TCN^{best}$ use a larger temporal context for classification than the DMD-based algorithms. The ratio of intervals reveals that adding a CNN front-end helps smoothing the classification in comparison to the isolated TCN. A CNN front-end can also reduce the number of parameters of the network with respect to an isolated TCN with the same hyper-parameters while improving its performance. Finally, we observe that the CNN front-end effectively works as a feature extractor that reduces the dimensionality of the input features and transforms them into consistent patterns that identify different combination of music and non-music sounds.

Chapter 6

SUMMARY AND FUTURE PERSPECTIVES

In Section 1.3, we set two goals: the development of state-of-the-art computational approaches to the task of relative music loudness estimation, and the introduction and promotion of the relative music loudness estimation task in the research field of MIR. In that same section, we also provided a summary of the contributions of this thesis. In Section 6.1, we extend the explanation of these contributions confirming that we have accomplished the goals. After this, in Section 6.2 we list the outcomes of the thesis that are publicly available, and finally, in Section 6.3, we provide several ideas for future research on the task of relative music loudness estimation.

6.1 Summary and impact of the contributions

In this section, we provide an extended description of the contributions of our doctoral work, which were already introduced in Section 1.3, including also an explanation of their impact on BMAT and academia. We consider that with these contributions we achieve the goals that we established for this thesis.

Introduction of the relative music loudness estimation task as a new MIR task. Under the current copyright management business model, the taxes that TV broadcasters pay to the corresponding copyright management organization depends on the percentage of music they broadcast and the proportion of this music that is played in the foreground and in the background. Currently, BMAT's relative music loudness estimation service continuously monitors around 4300 radio stations and TV channels to automatically detect the presence of music, and to classify it as foreground or background music. We considered that the industrial relevance of the task is sufficient reason to introduce it into the MIR research field. We have done so in this thesis through the publication of a dataset, the development of state-of-the-art computational approaches, and our participation in the MIREX competition. This has already fostered technological advances that can potentially benefit BMAT in the future such as the work by Jia et al. (2020), where the authors use OpenBMAT and the results of the MIREX competition.

Review of the literature and the available resources related to the detection of music. Even though there is no previous research about the task of relative music loudness estimation, given its similarity to the task of music detection, we can take advantage of much of the literature involving the detection of music, and the available resources related to

this task. With regards to the literature, we have provided a thorough review of the published approaches for the task of music detection and the tasks that combine the detection of music with the detection of non-music sounds such as speech and environmental sounds. As far as the available resources are concerned, we have described the public and private datasets for these tasks, and the annotation tools that can be used to annotate them. We have listed several shortcomings of the current available datasets that should be fixed for future datasets.

Development of an audio-events annotation tool with a focus on the annotation of their partial loudness. We have created BAT: an open-source, web-based tool for the annotation of audio events that provides a reliable way to annotate the partial loudness of these events when two or more of them overlap. It also enables the definition of different taxonomies to adapt to multiple tasks, and the possibility of cross-annotating audio data. Moreover, it is easy to deploy in servers thanks to the usage of Docker. BAT has proven to be functional and useful to BMAT, having provided hundreds of hours of annotations including the cross-annotation of OpenBMAT, and the annotation of the training datasets of the Deep Music Detector as well as other algorithms designed for a variety of tasks. In addition, it has been used in several one-off projects related to the study of the amount of music played on national and international radio stations and TV channels.

Creation and annotation of a private dataset for the task of relative music loudness estimation. This dataset was extracted from BMAT's private database and comprises approximately 44 hours of audio from TV channels and radio stations from all over the world distributed in 1322 two-minute audio files containing highly diverse broadcast sce-

narios. These audio files have been annotated by a single annotator using a taxonomy of four classes: *Music*, *Speech*, *Sound effects* and *Audience*. The annotations also include information about the partial loudness of these types of audio events where they overlap. In the dataset, 35% of the music appears isolated and 33% mixed with non-music sounds. The percentage of audio files that include class changes is 85%. We used this dataset to train the Deep Music Detector.

Creation and annotation of a public dataset for the tasks of relative music loudness estimation and music detection. Similarly to the private dataset, we sampled BMAT's private databse a second time to generate OpenBMAT: a public dataset containing around 27 hours of audio with annotations about the presence of music and its relative loudness with respect to simultaneous non-music sounds. The dataset is divided into 1647 one-minute audio files from well-known TV channels in France, Germany, Spain and the United Kingdom, and belonging to one of eight program types: children programs, documentaries, entertainment programs, music programs, news broadcasts, series and films, sport programs, and talk shows. These audio files have been cross-annotated obtaining high partial and full agreement levels for the relative music loudness estimation and the music detection tasks. According to these annotations, more than 50% of the audio files include class changes for the task of music detection, and around 66% include class changes for the task of relative music loudness estimation. 35% of OpenBMAT's content is background music at different volumes and mixed with a large variety of non-music sounds resulting from the diversity of broadcast contexts that we have incorporated in the dataset. With these characteristics we cover all the dataset shortcomings that we mentioned in Section 2.2.1.2, and

provide much needed public data, enabling future evaluations to be more transparent, reproducible and comparable. The dataset has already been downloaded 60 unique times[1] from Zenodo[2], and used by Jia et al. (2020), Gimeno et al. (2020) and ourselves (Meléndez-Catalán et al., 2020) in published papers.

Development of BMAT's new relative music loudness estimation algorithm. The main industrial achievement of this our doctoral work is to have provided BMAT with a new relative music loudness estimation algorithm. Using the private dataset described above, we trained the Deep Music Detector. This is the algorithm that is currently being used in production at BMAT, and which provides significantly better performance than its SVM predecessor. It also represents BMAT's first attempt at using deep learning. The core parts of the Deep Music Detector are implemented in C++ and make use of TensorFlow's C API, which results in a extremely fast computation time of 20 seconds per hour of audio using a single thread running on a CPU, and with a RAM memory peak of approximately 54 MB. This allows the Deep Music Detector to continuously monitor more than 4300 radio stations and TV channels. The Deep Music Detector has also been used by Jia et al. (2020) as a baseline for their relative music loudness estimation algorithm.

Development of state-of-the-art deep learning computational approaches for the estimation of music's relative loudness. We have proposed the usage of TCNs for the task of relative music loudness estimation. We use them in isolation but also in combination with a CNN front-end, which results in a novel deep learning architecture that we have

[1] last check on November 27th, 2020

[2] `https://zenodo.org/record/3381249`

named CNN-TCN, and that is inspired by the usage of CNNs in concatenation with RNNs to improve the performance of the RNNs. This new type of network uses fewer parameters than the isolated TCN, while boosting its performance, providing a smoother classification, and functioning as a feature extractor that produces consistent patterns identifying different combinations of music and non-music sounds. This novel architecture also exceeds the performance of the Deep Music Detector versions presented to the MIREX competition.

Organization of the MIREX competition as captain of the tasks of relative music loudness estimation, music detection, and speech detection. In 2018 and 2019, we organized the evaluation of these tasks providing evaluation datasets, running the submitted algorithms, and reporting their results. We also participated as authors in the tasks of relative music loudness estimation and music detection with two versions of the Deep Music Detector algorithm: *DMDv1* and *DMDv2*. *DMDv1* won the competition in 2018 outperforming the algorithms of four other participants for the task of music detection. It was also the first algorithm to participate in the relative music loudness estimation task. In 2019, *DMDv1* placed third in both tasks, after *DMDv2*. The first place was claimed by a CNN-TCN prototype that we produced during the elaboration of our third publication (Meléndez-Catalán et al., 2020).

6.2 List of publications

In this section, we list every outcome of our doctoral work that is publicly available. This includes the published papers, the developed resources for research, and the outcomes of our collaboration in MIREX competitions.

6.2.1 Papers

- **First accepted paper:** Meléndez-Catalán, B., Molina, E., and Gómez, E. (2017). BAT: an open-source, web-based audio-events annotation tool. In *3rd Web Audio Conference*.

- **Second accepted paper:** Meléndez-Catalán, B., Molina, E., and Gómez, E. (2019b). Open broadcast media audio from tv: A dataset of tv broadcast audio with relative music loudness annotations. *Transactions of the International Society for Music Information Retrieval*, 2(1):43–51.

- **Third accepted paper:** Meléndez-Catalán, B., Molina, E., and Gómez, E. (2020). Relative music loudness estimation using temporal convolutional networks and a cnn feature extraction front-end. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, volume 5, pages 273–280.

6.2.2 Research resources

- **BMAT's Annotation Tool (BAT):** `https://github.com/BlaiMelendezCatalan/BAT`

- **Open Broadcast Media Audio from TVs (OpenBMAT):** `https://zenodo.org/record/3381249`

6.2.3 MIREX collaboration

- **MIREX 2018 submission:** Meléndez-Catalán, B., Molina, E., and Gómez, E. (2018) Music and/or speech detection submission. *Mu-*

sic Information Retrieval Evaluation eX-change (MIREX).

- **MIREX 2018 evaluation results:** `https://www.music-ir.org/mirex/wiki/2018:Music_and_or_Speech_Detection_Results`
- **MIREX 2019 submission:** Meléndez-Catalán, B., Molina, E., and Gómez, E. (2019a) Music detection submission. *Music Information Retrieval Evaluation eX-change (MIREX).*
- **MIREX 2019 evaluation results:** `https://www.music-ir.org/mirex/wiki/2019:Music_Detection_Results`

6.3 Future perspectives

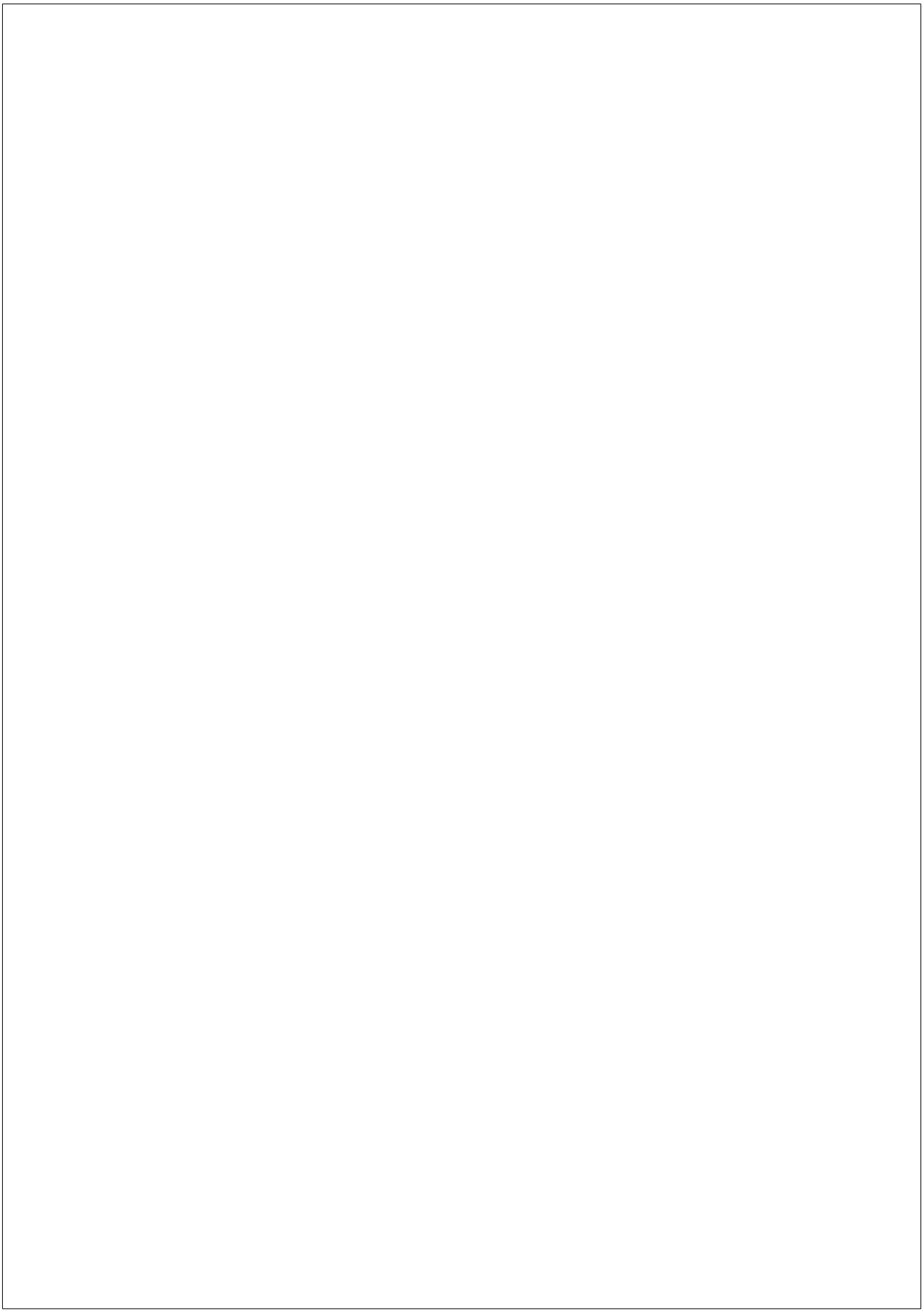
This PhD thesis comes to an end, but at BMAT we will carry on with our research on the task of relative music loudness estimation. There are multiple ways in which we can still improve our technology; in this section, we describe four of them:

- **Introducing the CNN-TCN architecture:** the first idea is to apply the results of our third publication and replace the CNN that currently constitutes the Deep Music Detector with a CNN-TCN architecture. Everything seems to indicate that this change can lead to an improvement of performance, but we need to make sure that the computation time of this new architecture will not represent a setback to put it into production.
- **Using audio in stereo:** at BMAT, we have started using recordings in stereo to improve our fingerprinting technology. We consider

that the Deep Music Detector can also benefit from the extra information that stereo audio files provide. As mentioned in Section 3.4, we already have an annotated dataset containing stereo audio. One idea could be to increase the number of channels of the Deep Music Detector's CNN from one to two in a similar way as is done for the RGB channels when working with images.

- **Implementing a multi-model Deep Music Detector:** some of our departments have spotted certain errors of the Deep Music Detector that are specific to channels with a certain type of content. We are already experimenting with a multi-model version of the algorithm. This version will contain several models derived from the original one, but fine-tuned to a particular type of content.

- **Using synthetic data:** all the datasets that we have used in our doctoral work contain real data. Using real data, we can be sure that the networks that we train are learning to model information that is representative of a part of the real broadcast audio. However, the annotations that we can generate for real data are affected by the subjectivity of the annotators and the shortcomings of the annotation method. The alternative to real data is synthetic data. Synthetic data would allow us to accurately generate different types of ground truth, which would broaden the range of network architectures that can be used for the task of relative music loudness estimation.



Bibliography

American National Standards Institute (1973). American national psychoacoustical terminology s3. 20.

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Bartsch, M. A. and Wakefield, G. H. (2005). Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on multimedia*, 7(1):96–104.

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Boersma, P. (2001). PRAAT, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345.

Bogdanov, D., Wack, N., Gómez Gutiérrez, E., Gulati, S., Boyer, H., Mayor, O., Roma Trepat, G., Salamon, J., Zapata González, J. R., Serra, X., et al. (2013). Essentia: An audio analysis library for music

information retrieval. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 493–498.

Butko, T. and Nadeu, C. (2011). Audio segmentation of broadcast news in the albayzin-2010 evaluation: overview, results, and discussion. *EURASIP Journal on Audio, Speech, and Music Processing*, 2011(1):1–10.

Cannam, C., Landone, C., Sandler, M., and Bello, J. P. (2006). The Sonic Visualizer: A Visualization Platform for Semantic Descriptors from. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR-06)*, pages 324–327.

Cartwright, M., Salamon, J., Seals, A., Nov, O., and Bello, J. P. (2018). Investigating the effect of sound-event loudness on crowdsourced audio annotations. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 341–345.

Cartwright, M., Seals, A., Salamon, J., Williams, A., Mikloska, S., MacConnell, D., Law, E., Bello, J. P., and Nov, O. (2017). Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations. volume 1, pages 1–21. ACM New York, NY, USA.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Choi, M., Lee, J., and Nam, J. (2018). Hybrid features for music and

speech detection. *Music Information Retrieval Evaluation eXchange (MIREX)*.

Coates, A. and Ng, A. Y. (2012). Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

de Benito-Gorron, D., Lozano-Diez, A., Toledano, D. T., and Gonzalez-Rodriguez, J. (2019). Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):9.

Doukhan, D. and Carrive, J. (2017). Investigating the use of semi-supervised convolutional neural network models for speech/music classification and segmentation. In *The Ninth International Conferences on Advances in Multimedia (MMEDIA)*.

Doukhan, D., Lechapt, E., Evrard, M., and Carrive, J. (2018). Ina's mirex 2018 music and speech detection system. *Music Information Retrieval Evaluation eXchange (MIREX)*.

Galliano, S., Geoffrois, E., Mostefa, D., Choukri, K., Bonastre, J.-F., and Gravier, G. (2005). The ester phase ii evaluation campaign for the rich transcription of french broadcast news. In *Ninth European Conference on Speech Communication and Technology*.

Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and 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)*, pages 776–780. IEEE.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Gfeller, B., Guo, R., Kilgour, K., Kumar, S., Lyon, J., Odell, J., Ritter, M., Roblek, D., Sharifi, M., Velimirović, M., et al. (2017). Now playing: Continuous low-power music recognition. In *NIPS 2017 Workshop: Machine Learning on the Phone (NIPS)*.

Giannakopoulos, T., Pikrakis, A., and Theodoridis, S. (2008). Music tracking in audio streams from movies. In *Proceedings of the IEEE 10th Workshop on Multimedia Signal Processing (MMSP)*, pages 950–955.

Gimeno, P., Mingote, V., Ortega, A., Miguel, A., and Lleida, E. (2020). Partial auc optimisation using recurrent neural networks for music detection with limited training data. *Proc. Interspeech 2020*, pages 3067–3071.

Gimeno, P., Viñals, I., Ortega, A., Miguel, A., and Lleida, E. (2018). A recurrent neural network approach to audio segmentation for broadcast domain data. In *IberSPEECH*, pages 87–91.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Gravier, G., Bonastre, J.-F., Geoffrois, E., Galliano, S., McTait, K., and Choukri, K. (2004). The ester evaluation campaign for the rich transcription of french broadcast news. In *LREC*.

Han, J. and Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Herrera, P., Massaguer, J., Cano, P., Gouyon, F., Koppenberger, M., Wack, N., and Fabra, U. P. (2005). Mucosa: a music content semantic annotator. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05)*.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Houtgast, T. and Steeneken, H. J. M. (1973). The modulation transfer function in room acoustics as a predictor of speech intelligibility. *Acta Acustica United with Acustica*, 28(1):66–73.

Izumitani, T., Mukai, R., and Kashino, K. (2008). A background music detection method based on robust feature extraction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13–16.

Jang, B.-Y., Heo, W.-H., Kim, J., and Kwon, O.-W. (2018). Music and/or speech detection methods for mirex 2018.

Jang, B.-Y., Heo, W.-H., Kim, J.-H., and Kwon, O.-W. (2019). Music detection from broadcast contents using convolutional neural networks with a mel-scale kernel. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):11.

Jia, B., Lv, J., Peng, X., Chen, Y., and Yang, S. (2020). Hierarchical regulated iterative network for joint task of music detection and music relative loudness estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Kiefer, J., Wolfowitz, J., et al. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krijnders, D. and Andringa, T. (2009). Soundscape annotation and environmental source recognition experiments in Assen (NL). *Inter Noise*.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 156–165.

Lemaire, Q. and Holzapfel, A. (2019). Temporal convolutional networks for speech and music detection in radio broadcast. In *Proceedings of*

the International Conference on Music Information Retrieval (ISMIR), pages 229–236.

Liu, C., Xie, L., Meng, H., et al. (2007). Classification of music and speech in mandarin news broadcasts. In *Proc. of the 9th Nat. Conf. on Man-Machine Speech Communication (NCMMSC), Huangshan, Anhui, China*.

Lu, L., Jiang, H., and Zhang, H. (2001). A robust audio classification and segmentation method. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 203–211.

Marolt, M. (2015). Music/speech classification and detection submission for mirex 2018. *Music Information Retrieval Evaluation eXchange (MIREX)*.

Meléndez-Catalán, B., Molina, E., and Gómez, E. (2017). BAT: an open-source, web-based audio events annotation tool. In *3rd Web Audio Conference*.

Meléndez-Catalán, B., Molina, E., and Gómez, E. (2019). Open broadcast media audio from tv: A dataset of tv broadcast audio with relative music loudness annotations. *Transactions of the International Society for Music Information Retrieval*, 2(1):43–51.

Meléndez-Catalán, B., Molina, E., and Gómez, E. (2020). Relative music loudness estimation using temporal convolutional networks and a cnn feature extraction front-end. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, volume 5, pages 273–280.

Mesaros, A., Heittola, T., and Virtanen, T. (2016). Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162.

Panagiotakis, C. and Tziritas, G. (2005). A speech/music discriminator based on RMS and zero-crossings. In *IEEE Transactions on Multimedia*, volume 7, pages 155–166.

Piczak, K. J. (2015). Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018.

Richard, G., Ramona, M., and Essid, S. (2007). Combined supervised and unsupervised approaches for automatic segmentation of radiophonic audio streams. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 461–464.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Rumelhart, D. E. and McClelland, J. L. (1987). *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pages 194–281.

Saunders, J. (1996). Real-time discrimination of broadcast speech/music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 993–996.

Scharf, B. (1964). Partial masking. *Acta Acustica united with Acustica*, 14(1):16–23.

Scharf, B. (1978). Loudness. *Handbook of perception*, 4:187–242.

Scheirer, E. and Slaney, M. (1997). Construction and evaluation of a robust multifeature speech/music discriminator. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1331–1334.

Schlüter, J. and Sonnleitner, R. (2012). Unsupervised feature learning for speech and music detection in radio broadcasts. In *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx-12)*.

Seyerlehner, K., Pohle, T., Schedl, M., and Widmer, G. (2007). Automatic music detection in television productions. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07)*.

Sjölander, K. and Beskow, J. (2000). Wavesurfer - an open source speech tool. In *INTERSPEECH*, volume 4, pages 464–467.

Snyder, D., Chen, G., and Povey, D. (2015). MUSAN: A Music, Speech, and Noise Corpus. arXiv:1510.08484v1.

Tsipas, N., Vrysis, L., Dimoulas, C., and Papanikolaou, G. (2017). Efficient audio-driven multimedia indexing through similarity-based speech/music discrimination. *Multimedia Tools and Applications*, 76(24):25603–25621.

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., and Sloetjes, H. (2006). ELAN: A professional framework for multimodality research. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 1556–1559.

Zhou, Y.-T. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *ICNN*, pages 71–78.

Zhu, Y., Sun, Q., and Rahardja, S. (2006). Detecting musical sounds in broadcast audio based on pitch tuning analysis. In *IEEE International Conference on Multimedia and Expo*, pages 13–16.

Zwicker, E. and Fastl, H. (1999). *Psychoacoustics*–facts and models.