THÈSE DE DOCTORAT
DE LA SORBONNE UNIVERSITÉ

Spécialité Informatique
École doctorale Informatique, Télécommunications et Électronique (Paris)
Sciences & Technologies de la Musique et du Son (UMR 9912)

MULTIMODAL ANALYSIS: Informed content estimation and audio source separation

par Gabriel MESEGUER BROCAL

dirigée par Geoffroy PEETERS

Soutenue publiquement le 7 juillet 2020 devant le jury composé de

Rapporteurs	Dr. Laurent Girin	Grenoble-INP - Institut Polytechnique de Grenoble
	Dr. Gael Richard	LTCI - Télécom Paris - Institut Polytechnique de Paris
Examinateurs	Dr. Rachel Bittner	Spotify New York
	Dr. Elena Cabrio	Université Côte d'Azur - Inria - CNRS - I3S
	Dr. Bruno Gas	ISIR - UMR7222 - Sorbonne Université Paris
	Dr. Perfecto Herrera Boyer	MTG - Universitat Pompeu Fabra Barcelona
	Dr. Antoine Liutkus	Centre Inria Nancy - Grand Est
Directeur	Dr. Geoffroy Peeters	LTCI - Télécom Paris - Institut Polytechnique de Paris

Abstract

This dissertation proposes the study of multimodal learning in the context of musical signals. Throughout, we focus on the interaction between audio signals and text information. Among the many text sources related to music that can be used (e.g. reviews, metadata, or social network feedback), we concentrate on lyrics. The singing voice directly connects the audio signal and the text information in a unique way, combining melody and lyrics where a linguistic dimension complements the abstraction of musical instruments. Our study focuses on the audio and lyrics interaction for targeting source separation and informed content estimation.

Real-world stimuli are produced by complex phenomena and their constant interaction in various domains. Our understanding learns useful abstractions that fuse different modalities into a joint representation. Multimodal learning describes methods that analyse phenomena from different modalities and their interaction in order to tackle complex tasks. This results in better and richer representations that improve the performance of the current machine learning methods.

To develop our multimodal analysis, we need first to address the lack of data containing singing voice with aligned lyrics. This data is mandatory to develop our ideas. Therefore, we investigate how to create such a dataset automatically leveraging resources from the World Wide Web. Creating this type of dataset is a challenge in itself that raises many research questions. We are constantly working with the classic "chicken or the egg" problem: acquiring and cleaning this data requires accurate models, but it is difficult to train models without data. We propose to use the teacher-student paradigm to develop a method where dataset creation and model learning are not seen as independent tasks but rather as complementary efforts. In this process, non-expert karaoke time-aligned lyrics and notes describe the lyrics as a sequence of time-aligned notes with their associated textual information. We then link each annotation to the correct audio and globally align the annotations to it. For this purpose, we use the normalized cross-correlation between the voice annotation sequence and the singing voice probability vector automatically, which is obtained using a deep convolutional neural network. Using the collected data we progressively improve that model. Every time we have an improved version, we can in turn correct and enhance the data.

Collecting data from the Internet comes with a price and it is error-prone. We propose a novel

data cleansing (a well-studied topic for cleaning erroneous labels in datasets) to identify automatically any errors which remain, allowing us to estimate the overall accuracy of the dataset, select points that are correct, and improve erroneous data. Our model is trained by automatically contrasting likely correct label pairs against local deformations of them. We demonstrate that the accuracy of a transcription model improves greatly when trained on filtered data with our proposed strategy compared with the accuracy when trained using the original dataset. After developing the dataset, we center our efforts in exploring the interaction between lyrics and audio in two different tasks.

First, we improve lyric segmentation by combining lyrics and audio using a model-agnostic early fusion approach. As a pre-processing step, we create a coordinate representation as self-similarity matrices (SMMs) of the same dimensions for both domains. This allows us to easy adapt an existing deep neural model to capture the structure of both domains. Through experiments, we show that each domain captures complementary information that benefit the overall performance.

Secondly, we explore the problem of music source separation (i.e. to isolate the different instruments that appear in an audio mixture) using conditioned learning. In this paradigm, we aim to effectively control data-driven models by context information. We present a novel approach based on the U-Net that implements conditioned learning using Feature-wise Linear Modulation (FiLM). We first formalise the problem as a multitask source separation using weak conditioning. In this scenario, our method performs several instrument separations with a single model without losing performance, adding just a small number of parameters. This shows that we can effectively control a generic neural network with some external information. We then hypothesize that knowing the aligned phonetic information is beneficial for the vocal separation task and investigate how we can integrate conditioning mechanisms into informed-source separation using strong conditioning. We adapt the FiLM technique for improving vocal source separation once we know the aligned phonetic sequence. We show that our strategy outperforms the standard non-conditioned architecture.

Finally, we summarise our contributions highlighting the main research questions we approach and our proposed answers. We discuss in detail potential future work, addressing each task individually. We propose new use cases of our dataset as well as ways of improving its reliability, and analyze our conditional approach and the different strategies to improve it.

Acknowledgments

This thesis is dedicated to Elena for her unconditional love and support every step of our way. 12.

I have a lot of many amazing people to thank. Without them I would not have gotten to this point, their presence have made this possible. I apologise in advance to those who I am forgetting. Sorry and thank you!

First of all, to my family. My parents, brother, and sister, who have always provided me unwavering support. You have brought me to where I am.

To all my professors who have guided me from the beginning of this journey. Especially to José Manuel Iñesta, Perfecto Herrera, Frédéric Bevilacqua and foremost Geoffroy Peeters for giving me the opportunity of doing a Ph.D and helping me during these years.

To my friends around the world from the different epochs of my life -Jona, Juan, Santi, Daps, Rob, Jakab, Filippo, Felipe, Derek, Matias, Martin, Magda, Tom, Phil, Adriana, Benjamin, Joseph, and Gino- because it does not matter when or where.

To the wonderful ISMIR community and all colleagues I have made there -Moha, Magdalena, Uri, Jordi, Delia, Helena, Zafar, Umut, and Javier-. It is a pleasure to constantly learn from you.

To my Spotify colleagues -Nicola, Simon, Keunwoo, Brian, Juanjo, Marco, Vincent, Ching, Andreas, David and Sebastian- for the stimulating experience of working with them. Especially to Rachel, being in this paragraph does not exclude you from the previous ones.

To IRCAM for being a unique place, to all the people who work there, and to the amazing researchers I had the pleasure to discuss with -Jordan, Dogac, Pierre, Philippe, Leo, Luc, Mathieu, Daniel, and Hugo- and my close friends -Guillaume, Alice, Hadrien and Aurelien- for lunches, dinners, drinks, translations, share hardships, and many other non-tangible things.

Finally, I am grateful to the French National Research Agency for providing the majority of funding for my PhD under the contract ANR-16-CE23-0017-01 (WASABI project).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Multimodal learning

Real word stimuli are usually produced by various simultaneously occurring complex phenomena each expressed in its own domain that constantly interact. Our understanding of these stimuli usually involves the fusion of different modalities into a joint representation. **Multimodal learning** aims at discovering the interaction between domains by developing methods to analyze phenomena from different modalities toward solving complex tasks. Formally, it is the discipline that studies how to use data from different domains/modalities that observe a common phenomenon toward learning/resolving complex tasks (Ramachandram and Taylor, 2017; Baltrušaitis et al., 2018). Multimodal learning gives the possibility to capture patterns that are not visible when working with individual modalities on their own, consolidating heterogeneous and disconnected data from various domains, and gaining an in-depth understanding of natural phenomena. This produces more robust and richer representations to improve the performance of the current machine learning methods (Baltrušaitis et al., 2018). For instance, if we want to study a musical artist, we can analyze his music. However, we will not have a complete vision since we are missing other dimensions e.g. lyrics, scores, video clips, album reviews, or interviews, that contribute to our idea of what a musical artist is and complement its purely musical dimension.

However, this comes with a certain cost and complexity (Atrey et al., 2010). It is much harder to discover relationships across modalities than relationships among features in the same modality, each dimension is captured in a different way, resulting in totally different types of data and the modalities may be correlated or independent. While correlated domains should work in a complementary manner and independent domains have to provide additional cues, the interaction between dimensions is hardly ever linear but has complex relationships and involves different abstraction levels. Additionally, each modality might have a different relevance for accomplishing a particular task or there might be the absence of modalities at some instants resulting in missing values. For

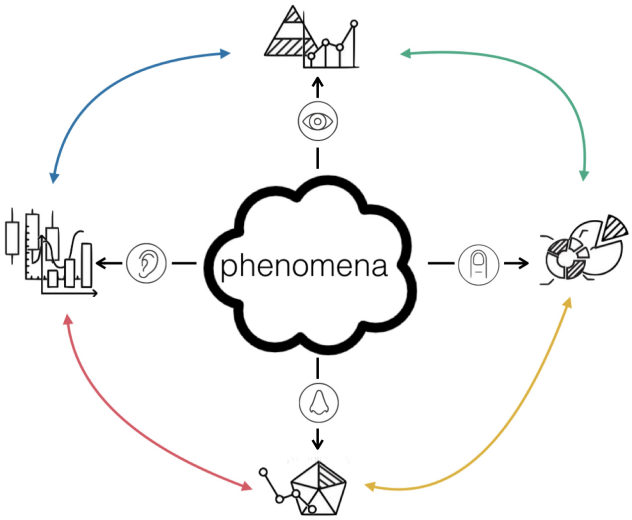


Figure 1.1: Our understanding of complex phenomena extracts information from different dimensions that constantly interact. For instance, we can process the same phenomena through the senses of smell, vision, ear, or touch. We extract different knowledge from each domain and combine them to improve our understanding of reality.

example, we can have a better understanding of a song if we analyze not only its audio and but also the comments of its creator (e.g. an interview). However, we first will need to identify the part of the interview where the artist talks about that song, extract the relevant information, learn how it aligns with the audio, and complements it. While a comment about the intention of a melody can be the key to understand the song, a simple anecdote about the recording session can be seen as noise.

A good multimodal learning model must satisfy certain properties that can be summarized in three main questions **how?**, **what?** and **when?** (Bengio et al., 2013; Baltrušaitis et al., 2018). These questions condition and define each multimodal learning approach.

1.1.1 How?

It refers to how the multimodal system is constructed i.e. defining the model and techniques used for designing the multimodal system. It explores how the knowledge learned from one modality can help another modality. There are two main approaches: model-agnostic and model-based. We can design models that are agnostic to the fact that the task is multimodal or being explicitly dependent on it, addressing the interaction between modalities in their construction. While in **model-agnostic** methods the multimodal learning is not directly dependent on a specific technique, **model-based** methods explicitly approaches the multimodality in its architecture design. Techniques that allow **model-based** architectures include 1) *multiple kernel learning (MKL)*, a support vector machines

(SVM) extension that use kernels for different modalities; 2) *graphical models* with generative models for joint probability (variations of hidden Markov models, dynamic Bayesian networks or Boltzmann Machines) and discriminative models for conditional probability (conditional random fields); and 3) *deep neural networks* for end-to-end training of multimodal representation with a wide variety of designs. Selecting one approach conditions how the following questions are addressed. Currently, *deep neural networks* are the most popular choice. Table 1.1 compares both **model-agnostic** and **model-based** methods.

Furthermore, we can also distinguish approaches regarding how the different modalities are used i.e. purely multimodal or contextual relation. In **purely multimodal** tasks, the different domains that look at the same phenomena work together to solve an external task. In **contextual** relation, one or more domains are used as 'context information' or guide to improve the representation of another. The context domains provide clues of *what* and *where* to 'look at'. This context information is an assistant but has a strong influence on the process. It helps to extract better content.

1.1.2 What?

It refers to what information is combined to have a better understanding of the phenomena. At a high level, it defines the sources of information to be used. Accessing to proper multimodal data is essential to carry on multimodal analysis. Currently, there is a renewed interest in **multimodal learning** thanks for the development of deep learning approaches. Since they require large training datasets to be successful, researchers have created several large multimodal annotated datasets (Bernardi et al., 2016). The most active communities (natural language processing and image processing) are those which use explicitly aligned datasets where there is a direct connection between sub-components of each modality. However, there is still a lack of labeled **multimodal** datasets for many multimodal tasks that hinder their growth. Creating multimodal datasets is a challenging task as it requires annotations which often are time-consuming and difficult to acquire.

On deeper levels, this question involves all the aspects related to data representation such as how to exploit correlations (complementary and contradictory elements), deal with different levels of noise, discover independency and redundancy, establish the confidence of each modality, or find intermodality and intramodality relationships. This is challenging due to the heterogeneity of multimodal data. We can compute multimodal representations either by considering each modality separately (each modality exists in its own space), but enforcing certain similarity or structure constraints to **coordinate** them, or by defining a **joint** representation that projects all the modalities into the same representation space. While **model-agnostic** approaches tend to create coordinate representations, **model-based** methods compute joint representations.

This question also concerns the **translation** challenges i.e. how to translate one modality to another. It is common to define an example-based dictionary where elements from different modalities are directly linked. This allows retrieving information from one modality given a query from

another. This idea is extended to models that can generate a translation between modalities. This goes beyond direct connections between elements and requires the ability to understand modalities to generate a new target sequence.

Table 1.1: A comparison between **model-agnostic** and **model-based** methods.

Model-based Multimodal Learning	**Model-agnostic Multimodal Learning**
Features are learned from data and can be shared within dimensions.	Features are manually designed and require prior knowledge about the underlying problem and data.
Implicit dimensionality reduction within architecture.	Feature selection and dimensionality reduction are often explicitly performed.
Have more flexiblilty for exploring different fusion types. Fusion architecture can be learned during training.	Typically performs early or late fusion. Rigid fusion architecture usually handcrafted.
Easily scalable in terms of data size and number of modalities.	Early fusion can be challenging and not scalable. Late-fusion rules may need to be defined.

1.1.3 When?

It refers to the problem of when to fuse the data. It can have two meanings:

1. **Horizontal - alignment:** it refers to which moment in the time should we fuse the different domains. It is related to **alignment** i.e. identifying the direct relations between subcomponents from different modalities. Imagine the audio of a song and its lyrics. Although the lyric gives information about the theme of the song, it does not say anything about the characteristics of the audio signal unless it is aligned in time with it. This aspect also deals with missing data or modality problems but also with synchronization issues (at the input and the output level) for dynamic processes. To tackle this challenge we need to measure similarities between different modalities and deal with possible long-range dependencies and ambiguities. While explicit alignment has a final goal of aligning sub-components between modalities, implicit alignment is used as an intermediate step for another task.

2. **Vertical - fusion:** at which depth of the system should we integrate the information from multiple modalities. It is one of the most researched aspects of multimodal machine learning. While **model-agnostic** methods fusion the different modalities independetly of the machine learning method, in **model-based** approaches, the fusion is dependent on the technique itself, explicitly addressing it in their construction.

    - **Model-agnostic** fusion methods include *early fusion - feature level* where features from the different domains are merged immediately after they are extracted, creating a higher

dimensional space; *late fusion - decision level* where each dimension is treated independently with parallel systems and the parallel decisions are merged to obtain the final decision; and *hybrid fusion* that combines early and late fusion techniques. Model-agnostic approaches can be implemented using almost any unimodal domain.

- **Model-based** approaches are designed to have multimodal fusion architecture at different depths of the process (Karpathy et al., 2014). Each multimodal learning problem defines its own architecture.

1.1.4 Multimodal tasks

Multimodal machine learning enables a wide range of applications, from human activity recognition and medical applications to autonomous systems and image and video description. The combination of image (or video) and text is one of the most common multimodal approaches. There is a large number of works that investigate captioning and description for both image (Bernardi et al., 2016; Vinyals et al., 2015; Xu et al., 2015; Kiros et al., 2014) and video (Venugopalan et al., 2014). Multimodal learning has been also used for retrieving images after providing a text description (Socher et al., 2014), reading lips into phrases (Chung et al., 2017), aligning books to movies to provide rich descriptive explanations (Zhu et al., 2015) or combining audio and video for speech recognition(Ngiam et al., 2011). Researchers also investigate the fusion of images and text into a joint representation (Srivastava and Salakhutdinov, 2014, 2012; Higgins et al., 2017). (Kaiser et al., 2017) proposed to train a single model that learns multiple task-specific encoders and decoders that combine images, audio, and text to perform image classification, image captioning, and machine translation. In audio, the most studied scenario is the fusion of audio and text for automatic speech recognition (ARS) (Graves et al., 2006) (to transcribe the audio signal into text) or for text-to-speech synthesis (van den Oord et al., 2016). Researchers have studied the co-occurrence of audio and visual events to train an audio network to correlate with visual (Aytar et al., 2016) or to find audio-visual correspondence task (Arandjelovic and Zisserman, 2017). We refer to (Ramachandram and Taylor, 2017), (Atrey et al., 2010) and (Baltrušaitis et al., 2018) for a detailed survey on the different multimodal approaches.

1.2 Multimodality in music

The most natural way to perceive music is through its acoustic rendering. However, through history music has been described and transmitted in several other different forms (Essid and Richard, 2012). Before being able to store it, music was materialized and exchanged as musical-scores. Since the growth of communication, music is essential to a wide range of disciplines. For instance, it conveys emotions in movies and becomes visual art in audiovisual installations or album cover designs. It is also widely described using text in editorial metadata or other social web content such as user-tags,

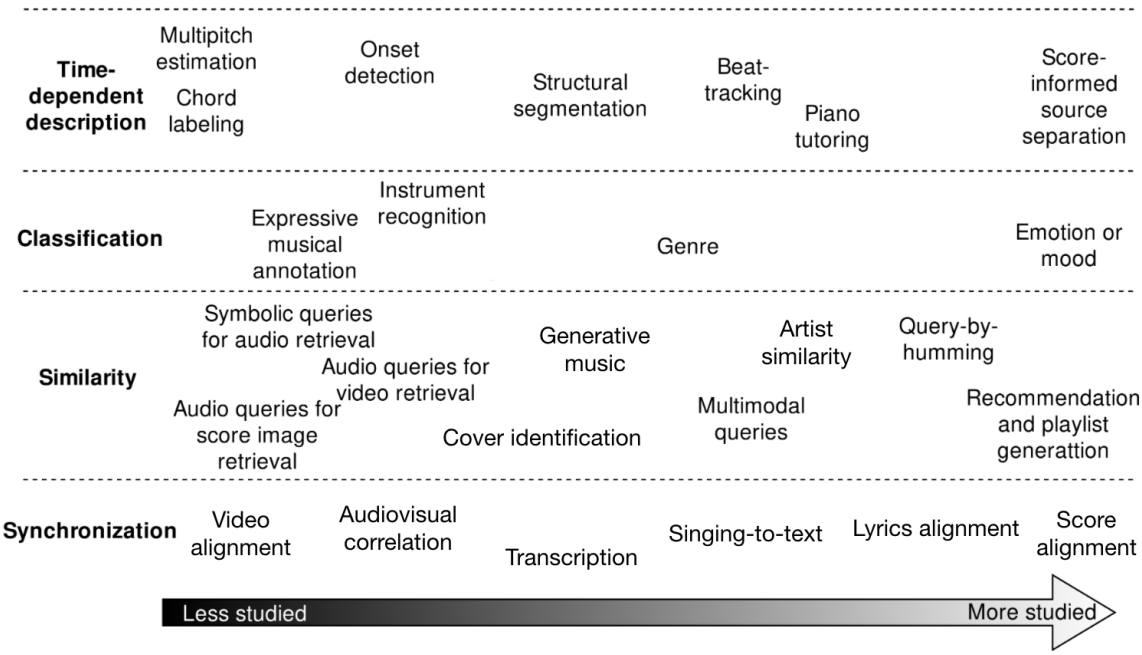


Figure 1.2: Outline of the different music multimodal tasks divided in 4 macro-tasks and plotted along a *less-more* studied axis. Figure reprinted and adapted from (Simonetta et al., 2019)

reviews, or ratings. We can even capture our mental perception of music (Gulluni et al., 2011). Not to mention that musicians have always performed music with motion and precise gestures. Even if it has always been an encouragement to consider music content beyond audio signals (Liem et al., 2011), it has been treated mostly only through its acoustic dimension, not benefiting profoundly from the other perspectives.

The field that studies these music audio signals is Music Information Retrieval (MIR) . MIR is an interdisciplinary research field dedicated to the understanding of music that combines theories, concepts, and techniques from music theory, computer science, signal processing perception, and cognition. Nowaday, researchers in MIR have a growing interest in understanding music through its various facets. Most of the studied music multimodal tasks fall into one of these four categories: **classification**, **similarity**, **synchronization**, and **time-dependent** representation (Simonetta et al., 2019).

Classification consists in assigning one or more labels to a song. Although also commonly studied as a single domain, mood and genre classification are one of the most addressed multimodal scenarios. The multimodel attempts hybridize audio and lyrics to exploit the complementary information between musical features and Latent Semantic Analysis (LSA) text topics (Laurier et al., 2008; Mayer et al., 2008). Currently, mood and genre are awakening a new interest in the community. While reviews or user-feedbacks are used for genre classificaiton (Oramas et al., 2017a), embedding

music and lyrics produces advantageous models for mood classificaiton (Su and Xue, 2017; Huang et al., 2016; Xue et al., 2015).

Similarity refers to methods that measure the similarity between the content of different modalities. It includes mostly retrieving documents through a query. Multimodality arises from the fact that the query may be from a different domain than the retrieved document, for instance, query-by-lyrics (Müller et al., 2007). Different domains can be combined to create better representations, e.g. artist similarity ranked from acoustic, semantic, and social view data (McFee and Lanckriet, 2011) or more complete descriptions via embedding spaces from biographies, audio signal and available feedback data (Oramas et al., 2017b). We can even think in mapping the audio into a representation in the mind of the musicologist for complex electro-acoustic music (Gulluni et al., 2011). Another interesting task is generating new information in one domain given a query from another, by means of 'generative models'. For instance when generating images from audio and generate audio from images with a deep generative adversarial network (Chen et al., 2017). The similarity between domains can also be implicitly modeled inside classification methods.

Synchronization tasks focus on the alignment in time or space between elements in different domains. Lyrics and score alignment are the most popular problems followed by singing-to-text and score transcription. Annotated chords progression modeled with Hidden Markov Models have been proved useful for improving lyric alignment (Mauch et al., 2010, 2012) or aligning the audio to them (McVicar et al., 2011). Audiovisual correlation in music videos defines semantic relationships between the stream of audio and video (Gillet et al., 2007). Note how synchronization can be a pre-step for defining more complex similarity relationships between domains or for solving classification problems. Additionally, some tasks such as singing-to-text and score transcription need a direct similarity measure of local elements to be solved.

Time-dependent tasks compute time-dependent descriptions of the music e.g. onset detection. The different domains are combined to enhance the description, for instance by using the video of the musician performing a piece to detect playing activity of the various instrument in multi-pitch estimation (Dinesh et al., 2017). Other examples are structure segmentation where we identify the music piece structure using video, lyrics, or scores (Zhu et al., 2005; Cheng et al., 2009a; Gregorio and Kim, 2016) or audiovisual drum transcription that exploits both modalities (Gillet and Richard, 2005) or using the video as event detection guide (McGuinness et al., 2007). Nevertheless, score-informed source separation is probably the most studied task Ewert et al. (2014). We guide the separation using a musical score which is often strongly correlated in time and frequency with music.

In the next chapters, we provide an exhaustive literature review for the multimodal tasks that are related to our work. For further details on music multimodal tasks and methods, we refer to (Essid and Richard, 2012) and (Simonetta et al., 2019).

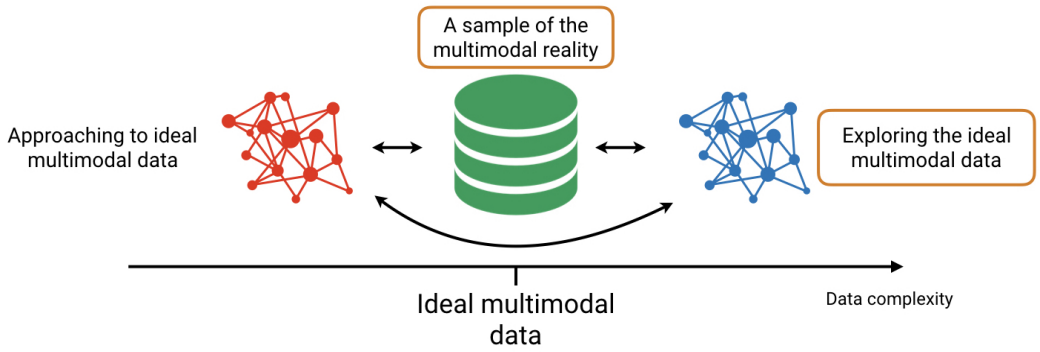


Figure 1.3: We center our efforts in creating a multimodal dataset that acts as a sample of the multimodal reality we investigate, audio with lyrics aligned in time. We can then use it in two different directions, for either developing models that can produce such type of multimodal data or exploring a multimodal formalization showing that this is beneficial to the performance of the models. We opt for the latter.

1.3 Problem formalization

Research on multimodality receives a growing interest in MIR. Multimodal MIR is an exciting field that tackles music problems more globally, exploiting the natural multidimensions of music. Nevertheless, it still grows at a much slower rate than other fields in the community. Existing multimodal music approaches and datasets are not standardized and researchers are still establishing the foundations of it.

In this dissertation, we explore a defined multimodal scenario, combining music audio and text information. Text can refer to many different textual sources: editorial reviews, social web content, user-tags, or ratings. Among all of them, we focus here on lyrics. In popular music, lyrics have a direct connection to the audio signal via the singing voice, which is one of the most salient components in a musical piece (Demetriou et al., 2018). The singing voice acts as a musical instrument and at the same time conveys semantic meaning through the lyrics (Humphrey et al., 2018). It is the central element around which songs are composed, defining the lead melody and creating relationships between sound and meaning, adding a linguistic dimension that complements the abstraction of the musical instruments. This connection tells stories and conveys emotions, improving our listening experience. Some musicians even accentuate this connection by composing music that reflects the literal meaning of lyrics e.g. descending scales would accompany lyrics about going down, or happy and energic music would accompany lyrics about joy. For these reasons, the singing voice is a very motivating and useful multimodal scenario.

Our goal is to develop methods that use both lyrics and audio information to improve downstream MIR tasks. Due to the relatively under-development of multimodal analysis in music and in order to tackle the different MIR tasks, we need first to address generic machine learning aspects. All machine learning problems have three core elements: the example **data** from which a system learns

generic patterns to solve the task, the **system** itself (with all its components e.g. optimization or losses), and the **evaluation** process to check that the system behaves as expected. Since the emergence of neural networks, there have been exponential advances in the representation capabilities of **systems**. But datasets and **evaluation** techniques have surprisingly grown at a much slower rate (Sun et al., 2017). Recenetly, fields like **active**, **weakly-supervised** or **semi-supervised** learning have appeared. Nevertheless, there is still an absent of large and good quality datasets for music multimodal analysis, limiting the development of new approaches. Hence, we first investigate how to automatically create a large and good quality dataset with lyrics and vocal notes aligned in time. The dataset is a sample of the multimodal reality we aim to investigate, but its automatic creation is often error-prone. We then tackle questions related to the **evaluation** and to the problem of both training and evaluating in the presence of label noise, proposing a *self-supervised* method to automatically identify possibly wrong labels.

Once the dataset is defined, it can be used in two different directions (see Figure 1.3). On one hand, we can use it to dig into tasks that can automatically transform the current data into the desired multimodal data, e.g. exploring tasks such as automatic lyrics alignment or singing voice transcription systems. On the other hand, we can investigate how to improve downstream MIR tasks, showing that a multimodal formalization that exploits the natural multiple dimensions of music is beneficial for the performance of the models. Finally, it can be used to train models that tackle both scenarios at the same time. In this thesis, we focus only on exploring how to improve MIR tasks once we have access to the aligned data. The two MIR tasks we study are: *structure segmentation* and *source separation*. To use the audio and the lyrics, we study the conditioning of models which allows to guide the resolution of a problem based on external information (see Chapter 8). Lastly and following the previous formalization, we define our multimodal analysis of lyrics and music as follows:

- **When?** to properly explore the relationship between the audio signal and its 'meaning' (lyrics), we need an explicit alignment between lyrics and the audio. We develop our dataset having in mind this goal: to obtain a large amount of songs with their lyrics aligned in time. Since *'when?'* can also refer to which moment in the learning process we are combining the lyrics and audio, we use **model-agnostic** fusion for structure segmentation (see Chapter 7) and **model-based** approaches to condition the singing voice source separation with respect to the phoneme information (see Chapter 9).

- **What?** during the course of our work we explore several directions to use lyrics and audio. We first transform the audio signal to highlight vocal areas for the creation of the dataset in an **agnostic** way (see Chapter 4) to adapt the lyrics alignment to the audio. We develop also a **joint** representation for structure segmentation and use the text information as prior knowledge (**context**) about the audio signal to condition a singing voice source separation model.

- **How?** The dataset creation belongs to the **semi-supervised**, **active** and **weakly** learning paradigms (see Chapter 4). Although having access to this kind of data opens the door to many generative methods (e.g. automatically generating lyrics given a particular melody), the selected MIR tasks, *structure segmentation* and *source separation*, are moslty studied in a supervised learning paradigm using discriminative models. Our main learning machines are deep neural networks due to their flexibility and ability to learn a shared representation (see Chapter 2).

1.4 Dissertation Summary and Contributions

At the start of this work, the question of how to approach a multimodal analysis of lyrics and audio remained open, and the current solutions study the use of both in a weakly aligned way. The recent success of data-driven methods in many MIR tasks and the renewed interest in multimodal analysis promise exciting times. The goal of building a data-driven approach to explore the rich interaction between lyrics and audio gives rise to the need for large amounts of annotated data. Despite the importance the singing voice has on how we enjoy music, there is a lack of large datasets of this kind and a relatively small amount of multimodal work applied to this task. This opens a challenging area of research. In this dissertation we address the following questions:

1. How can we obtain large amounts of labeled data where lyrics and its melodic representation are aligned in time with the audio to train data-driven methods?
2. How can we automatically identify and fix errors in these labels?
3. How can we exploit the inherent relationships between lyrics and audio to improve the performance for lyrics segmentation?
4. How can we effectively control data-driven models? Can we use prior knowledge about the audio signal defined by the lyrics to improve the isolation of the singing voice from the mixture?

The remainder of this dissertation is organized as follows. In Chapter 2, we give an overview of core tools used to develop our ideas, including further discussion about supervised learning and relevant concepts from deep neural networks. Chapter 3 provides a detailed description of our multimodal dataset with lyrics and vocal notes aligned in time at different levels of granularity. It also outlines the different versions and the characteristics of the data. Chapter 4 describes how we create the dataset where we explore Active learning and Weakly-supervised Learning techniques, creating an interaction between the dataset creation and model learning that benefits each other. In Chapter 5, we deepened into the labeling errors and propose automatic solutions to several types of issues. However, we cannot measure if the new labels are better or worse. In Chapter 6, we propose a novel data cleansing method for automatically knowing the current status of the dataset. Our

method exploits the local structure of the labels to find possible errors in vocal note event annotations. This chapter is the last chapter concerning our multimodal dataset. Chapter 7 explores lyrics segmentation as a first scenario to use text and audio, showing that they capture complementary structure. Chapter 8 provides a first approximation to conditioning learning for music source separation. We present there a novel approach for performing multitask source separation effectively controlling a generic neural network to perform several instruments isolations. Chapter 9 extends this approach to singing vocal source separation, using prior knowledge about the phonetic characteristics of the signal using strong conditioning to improve vocal separation. Finally, we conclude and give directions for future work in Chapter 10.

The multimodal analysis of lyrics and audio is without a doubt still an open question. We provide a new grain of sand here to help research grow this field. We develop dataset-focused strategies and contribute a new dataset. We also explore conditioning techniques that show that multimodal formalizations that exploit the natural multidimensionality of music help to solve problems satisfactorily. We are optimistic that this work will help future researchers to tackle this challenging topic with more resources and ideas.

Chapter 2

Tools

In this thesis we use a common collection of tools to develop our ideas. Broadly speaking, we employ techniques from the field of machine learning. In this chapter, we give a high-level overview of this field and a more specific definition of the various tools we will use.

2.1 Machine learning

Machine learning is a research discipline that designs methods for enabling computers to learn to do particular tasks without being explicitly programmed to do so. Instead of defining any custom algorithm with specific logic, machine learning methods learn its own logic from data, where they automatically discover the needed patterns to carry out the desired tasks (Goodfellow et al., 2016). Within machine learning, and based on the kind of data available, there are several ways of solving tasks. They differ in their availability of accessing prior knowledge of what the output of the model should be.

Supervised learning uses a collection of labeled data where we specify the desired output for a given input. Using the labeled data, we can directly evaluate the accuracy of a model e.i. measuring how correct the answers are (Goodfellow et al., 2016). However, the labels are not always available. **Unsupervised learning** stands as a solution to these cases. There, models use unlabelled data (that is, just the input) to infer the natural structure within the set of data points. Since we do not know the labels, most unsupervised learning methods have no specific way to measure model performances. **Semi-supervised learning** takes a middle ground between the two previous approaches and combines both, labeled and unlabelled data. It uses labeled datasets (usually smaller than unlabeled datasets) to extract knowledge, allowing to infer the labels of the larger unlabeled set. Finally, **reinforcement learning** trains models focusing on the optimal way of making decisions. In this paradigm, we provide feedback (rewards or penalties) for guiding models when they perform actions. In this thesis, we mainly use supervised learning along with a specific

semi-supervised learning method, called the teacher-student paradigm (see Chapter 4.5).

2.1.1 Supervised Learning

Supervised learning uses labeled data to discover functions that map input-output pairs. When we talked about labeled data we refer to the scenario in which each input value is tagged with the answer the model needs to find on its own. Thus, the learning process consists of identifying patterns in the input data that correlate with the desired target output. Given a set $\mathcal{S} = (x_1, y_1), (x_2, y_2), ..., (x_{|\mathcal{S}|}, y_{|\mathcal{S}|})$ of input-output pairs where $x_n \in \mathcal{X}$ is the input space and $y_n \in \mathcal{Y}$ the target space, we aim to find a function f_S with controllable parameters θ that capture the relationship between x and y so that:

$$f_S(x_i, \theta) = y'_i \approx y_i \tag{2.1}$$

where y'_i is the output of the model and y_i the real answer we aim to obtain. We assume that the training pairs (x_i, y_i) are drawn from an unknown joint probability distribution $p(x, y)$, of which we only know the training set $\mathcal{S}$. We approximate the probability distribution $p(x, y)$ with f by adjusting the parameters θ based solely on $\mathcal{S}$. The ultimate goal of a trained model / learned function $f_S(x_i, \theta)$ is to take new unseen data x_i (not in $\mathcal{S}$) and correctly determine $f_S(x_i, \theta) = y'_i \approx y_i$ based only on the prior knowledge acquired. We call this process inference or prediction. Note that the "correct" output is determined entirely during the training phase using only the data points of $\mathcal{S}$. It is frequent to assume that the pairs $(x_i, y_i) \in \mathcal{S}$ are true, meaning that the label y_i is the correct answer to x_i. Nonetheless, this often does not hold. Noisy and/or incorrect labels will certainly reduce the effectiveness of the model, and sometimes there is no clear-cut way to assign univocal labels (e.g., some chord or mood labels). We study this matter in detail on Chapter 6.

Because of we have access to the labeled data, we can directly evaluate the accuracy of a trained model. However, this does not necessarily reflect real-world performances since the data to which we have access may not contain all the cases we face in the real world. The error of a model can be broken down into three distinct parts. The first part is the irreducible error due to the noise in $p(x, y)$. This error is intrinsic to the phenomenon being modeled and cannot be eliminated through good modeling practices. The other two types of errors are related to the training dataset $\mathcal{S}$ and our model definition $f(\theta)$.

When creating $\mathcal{S}$, we want it to be a representative and well balanced (each class label is equally represented) description of the unknown joint probability distribution $p(x, y)$. Our datasets are "samples" taken from an unfathomable reality. Ideally, they would capture that reality in its essential aspects and guarantee good models. But our sampling techniques and limitations lead us to unrepresentative samples. As a result, $\mathcal{S}$ is often far from $p(x, y)$ and does not contain all the

possibles x nor the outputs y. This produces a variation in performance between the direct evaluation compute on $\mathcal{S}$ and the real-word performance. This difference is known as the *variance* error and measures the amount by which the performance may vary for the various sets we can draw from $p(x, y)$. It decreases augmenting the size and representativity of the training **data** $\mathcal{S}$. If we define the expected peformance of our selected function $f(\theta)$ across all possible draws from $p(x, y)$ as $E[f(\theta)]$ and $f_S(\theta)$ as the actual performance on $\mathcal{S}$, we can formalize the *variance* as:

$$Var_\theta = E[(f_S(\theta) - E[f(\theta)])^2] \tag{2.2}$$

Finally, there is always also the *bias* error related to a specific task independently of the training data $\mathcal{S}$. It refers to the constant inherent error to our particular formalization of the problem $f(\theta)$. The *bias* describes how far our selected $f(\theta)$ is from the ideal unknown function f^* that describes perfectly the joint probability distribution $p(x, y)$. It decreases augmenting the complexity of the **model** defined by the parameters θ. We can formalize *bias* as:

$$Bias_\theta = E[f(\theta)] - f^* \tag{2.3}$$

The goal of any supervised learning model is to achieve low *bias* and low *variance*. The proper level of model complexity θ is generally determined by the nature of $\mathcal{S}$. In an ideal scenario, we would be able to develop the perfect model using infinite training data, thereby eliminating all errors due to *bias* and *variance*. That is it, the training process will result in a well-approximate minimum over the unknown $p(x, y)$. Adjusting θ typically involves using a task-specific "loss function" $\mathcal{L}$ which measures the agreement between the output of the model y' and the target y, i.e. the error that measures the model performance whose output is y' with respect to the target output y. Instead of computing the loss function for a single example, we average it for many training examples (ideally the whole training set $\mathcal{S}$). This function is called "cost function". It is also usual to use the term "objective function" to refer to any function optimized during training. The choice of an objective function $\mathcal{L}$ depends on the characteristics of the target space $\mathcal{Y}$. We minimize $\mathcal{L}$ over the training set $\mathcal{S}$ by adjusting θ being the total error due to both, *bias* and *variance*. Unfortunately, we cannot directly calculate the contribution of each term (*bias* and *variance*) because we do not know the actual target joint probability distribution $p(x, y)$. Moreover, *bias* and *variance* typically move in opposite directions of each other in balance known as *bias-variance trade-off* (see Figure 2.1). In practice, we move between simple models (or with rigid underlying structure) that oversimplify the relationship between x and y, reducing *variance*, but potentially introducing *bias* known as "underfit"; to more complex models with reducing *bias*, but potentially introducing *variance* known as "overfit". Since current models tend to be complex (with a larger number of θ), the success of a model depends on $\mathcal{S}$. If it is small, or not uniformly spread throughout different possible scenarios, complex models will "overfit", i.e. learning a function that fits only $\mathcal{S}$ very well capturing randomness

in the data and going beyond the true signal into the noise, without learning the actual trend or structure in $p(x, y)$. This results in unnecessarily complicate the relationship between x and y and therefore tends to generalize poorly.

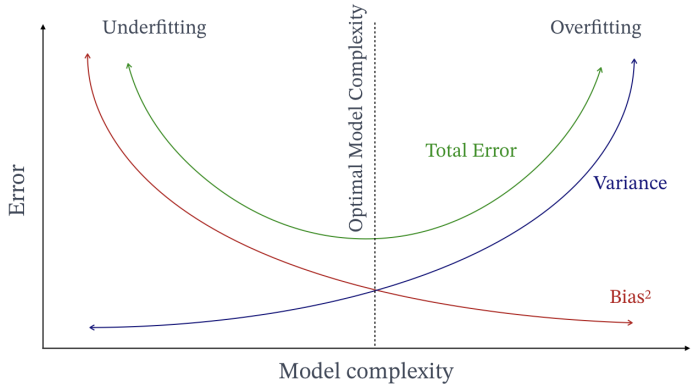


Figure 2.1: Bias-variance trade-off. Complex models with high *variance* learns a function that fits only the training set $\mathcal{S}$. Simple models with high *bias* oversimplify the relationship between x and y. Neither of these scenarios captures the actual structure in $p(x, y)$.

We measure the evolution of the *bias-variance trade-off* by dividing the training set $\mathcal{S}$ into three sets: training, validation, and test. The training set is the actual set used for training our model. The test set measures the expected performance of our model in the real-world. Ideally, we want our test set to a be different draw to $\mathcal{S}$ of $p(x, y)$, which will reflect a more precise performance. However, this is not always possible. The validation set is used for finding the optimal model complexity with the minimum error. We do so by comparing during training how the error evolves in the training set against the validation set. The training set increases the model complexity, i.e. minimizing the $bias^2$. Testing each version of our model in the validation set, we can have an approximation of the *variance*. Additionally, the validation set is also used for tuning the internal control parameters of our model.

2.1.2 Neural Networks

Most of the machine learning algorithms used in this thesis come from the Deep Neural Networks (DNN) class of models. DNN models break down and distribute tasks onto machine learning algorithms that are organized in consecutive layers built on the output from the previous layer. Loosely inspired by the brain (where the name 'neural network' arises) where neurons are associated one to another passing information, DNN algorithms consist of a sequence of non-linear processing stages passing information to each other. The basic unit is a "neurons" (see Figure 2.2):

$$h(x) = \sigma(Wx + b) \tag{2.4}$$

where σ is a non-linear activation function, $x \in \mathbb{R}^D$ is the input of the layer, $W \in \mathbb{R}^{1xD}$ the weight matrix, b the *bias* vector. The bias term b helps models to represent patterns that do not necessarily pass through the origin. The weights W perform a linear transformation of the input data x. Indeed, without a non-linear activation function, a DNN architecture is just a group of linear transformations. Both W and b are parameters that the model has to learn during the training process. The activation function σ is the key element and introduces non-linear properties to the network for learning complex relationships between input and output.

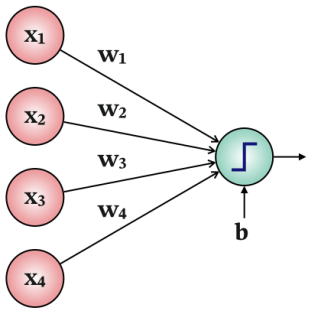


Figure 2.2: A neuron computes a vector to scalar operation and apply a non-linearity operation to the result.

DNN models are composed of layers. There are three main types of layers: the *input* layer that receives x, the *output* layer that generates y' and at least one *'hidden layer'*. *Hidden layers* have as input the output of another layer (not the original one x) and output also intermediate features (not the final output y'). *Hidden layers* are in charge of capturing complex relationships by progressively computing a more 'abstract' representation of the input x, i.e. the first layer detects a first abstraction of x, the second layer an abstraction of the first abstraction, the third layer abstraction of those abstractions, and so progressively. Each layer consists of a set of simply connected neurons that act in parallel (see Figure 2.3). Each neuron in a layer is connected to all neurons in the previous layer. It receives the inputs and computes its own activation value (a vector-to-scalar function), capturing a different input combination (Goodfellow et al., 2016). This makes each neuron independent to the rest of neurons of the same layer (they do not share any connections). This architecture is called *"fully-connected"*.

DNN models make predictions by "forward propagating" the input data through the network layer by layer to the final layer which outputs a prediction y'. The final prediction can be viewed as a long series of equations of the input. This process is known as **forward propagation**.

The variable θ defines the total number of parameters to learn. We adjust them by minimizing an objective function $\mathcal{L}$ using **Backpropagation** (Rumelhart et al., 1986). Backpropagation is the method that computes the gradient of $\mathcal{L}$, measuring the deviation between the network's output y' and the target output y with respect to θ. At the heart of backpropagation is an expression for

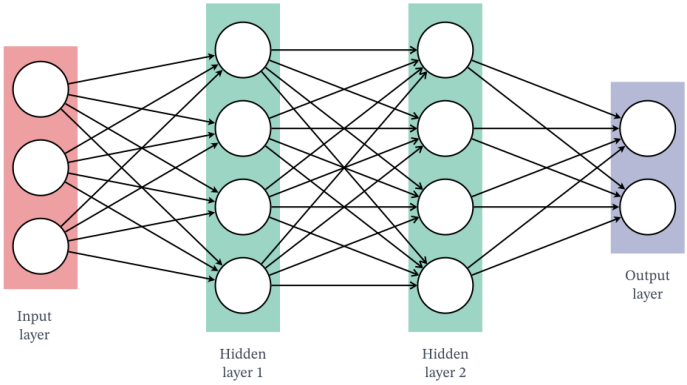


Figure 2.3: Deep neural networks consist of a set of layers that progressively transform the input information into the output data, discovering the necessary abstractions for solving a task.

the partial derivative $\frac{\partial \mathcal{L}}{\partial w}$ of the cost function $\mathcal{L}$ with respect to any parameter θ in the network. In other words, backpropagation tells how much the objective function $\mathcal{L}$ changes when a parameter changes, i.e. how the overall behavior of the net is affected by each parameter. Similar to forward propagation, the model error (differences between y' and y) is "back propagated" layer by layer through the output to the input layer updating each parameter progressively. Using the partial derivatives $\frac{\partial \mathcal{L}}{\partial w}$, we update θ using gradient descent methods. These methods minimize functions by iteratively moving in the error surface in the direction of steepest descent, as defined by the negative of the gradient down toward a minimum error value. The most utilized gradient descent method is stochastic gradient descent (SGD). At each training iteration, SGD updates each parameter by subtracting the gradient of the loss with respect to $\mathcal{L}(\theta_t)$, scaled by the **"learning rate"** η. The resulting product is called the gradient step in the error surface:

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(\theta_t) \tag{2.5}$$

Current $\mathcal{S}$ used for training DNN cannot be employed all at once. Instead, we train over a tiny subset called a **"minibatch"**. Minibatches are sampled randomly at each iteration of the gradient descent. The size of the **minibatch** (also named just as "batch size") conditions the **learning rate** value. The optimal value also depends on the morphology of $\mathcal{S}$. Minibatches cause the objective function to change stochastically at each iteration of optimization. If it is small and the learning rate large, we can move far from the desired minimum error value. On the other hand, it is is too small, we may never reach it.

When designing a DNN there are many choices to be made such as the number of layers (depth of the net) and the number of neurons in each one. The number of neurons defines how many different input combinations. The number of layers is connected with the capacity of the model to find hierarchical transformation with more and more abstract representations. Both are related to

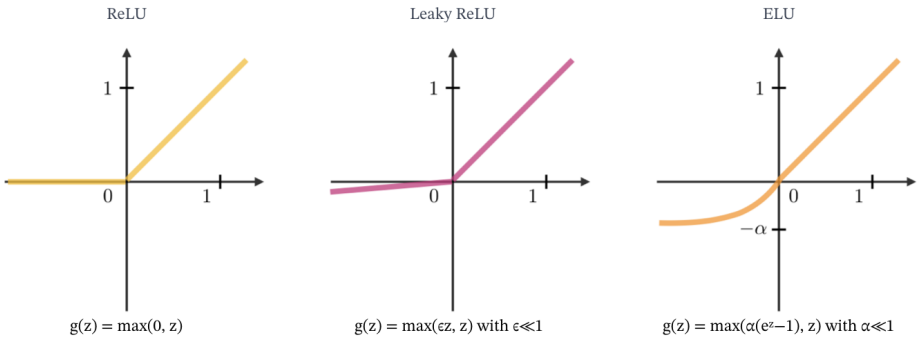


Figure 2.4: Rectifier non-linearities activation function. The rectified linear unit layer (**ReLU**) represents a nearly linear function with a threshold operation where values below zero are set to zero. **LeakyReLU** and **ELU** are alternatives with a non-zero derivate for negative values. Hence, we can backpropagate the error also in these values. Figure copyright CS 230 Deep learning.

the complexity of the model and the *bias-variance* trade-off. Increasing them increases modeling power but also exacerbates overfitting. To overcome the overfitting, it is usual to use regularization techniques that simplify the model. These factors penalize on the model's complexity to ensure that the optimized neural network's *variance* is not too high. Another important ingredient is the non-linearity σ applied at each neuron. This is essential for the model not to reproduce a linear combination of the inputs and discover complex relationships. There are many different functions such as logistic or hyperbolic tangent. However, the most usual choices are among rectifier non-linear functions (see Figure 2.4) due to its computational efficiency, i.e. its tendency to produce sparse representation and to reduce the vanishing gradient problem when the gradients of the loss function approaches zero, making the network hard to train (Nair and Hinton, 2010; Glorot et al., 2011).

Many of the concepts presented in this section uses the *fully-connected* architecture as an illustrative example are common to other DNN architectures.

2.1.3 Convolutional neural networks

The main limitation of *fully-connected* architectures is that they do not scale well. For instance, if we want to process an image of 64x64x3 (64 wide, 64 high and 3 color channels), we will need 12288 learnable weights for a single neuron. Furthermore, we are almost certain that we want to have many of such neurons to compute different combinations and several hidden layers to obtain complex relationships. As a result, we are increasing the complexity and the number of parameters θ of our model which would quickly lead to overfitting. Convolutional Neural Network (CNN) architectures are ordinary DNN that make the explicit assumption that the inputs are images (LeCun and Bengio, 1995). This allows efficient implementations, reducing vastly the number of parameters. CNN are also made up of neurons with learnable weights and biases, that perform a vector-to-scalar operation followed by a non-linearity activation. We also train them with an objective function $\mathcal{L}$ that expresses

a single differentiable score.

To constrain the architecture noticeably, CNN take advantage of the image characteristics. A digital image is a three dimensional function $i(w, h, d)$, where w, h and d are spatial coordinates. Any coordinate is usually called a 'pixel' and its amplitude $i(w, h, d)$ 'intensity'. Connectivity between pixels and spatial correlation (a pixel depends on both itself and its surrounding) are fundamental concepts that define the basic image components. These components make possible more complex attributes to finally create concepts such as a dog or a nose. In summary, pixel position and neighborhood have semantic meanings and elements of interest can appear anywhere in the image.

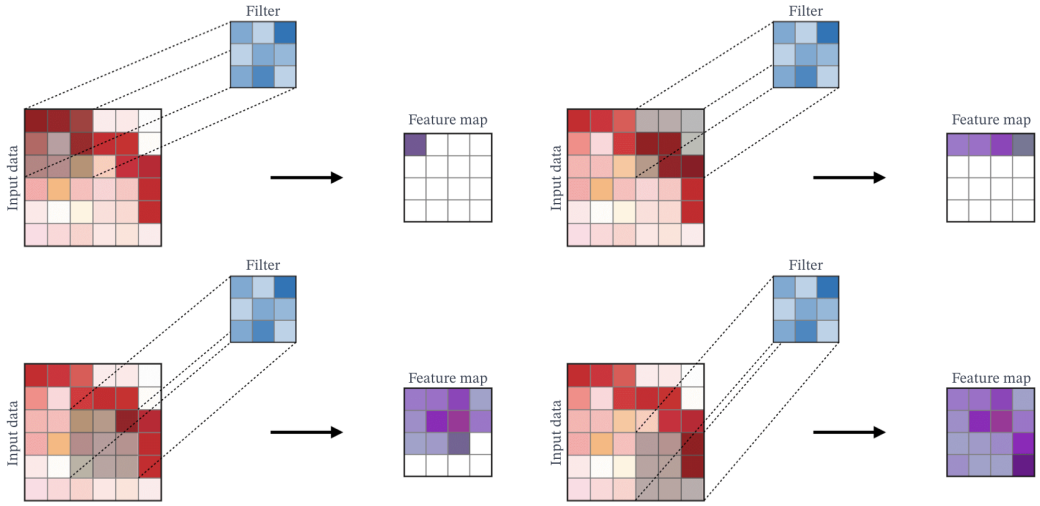


Figure 2.5: A convolution operation consists of aplying a filter that scans the input image with respect to its dimensions. Figure copyright CS 230 Deep learning.

The core concept of CNN is filtering. Filters (also called kernels) have been used since the foundations of the image processing domain. They are in charge of detecting image attributes, defining in which locations they occur and how strongly they seem to appear. We apply a filter over the whole image using a convolution operation (see Figure 2.5). When convolving a filter, we slide it over the whole image. At each location, we compute an element-wise multiplication between each filter element and the input elements it overlaps, summing up the result to obtain the output in the current filter location. As a result, we obtain a matrix that captures the activations of the filter over the whole image, i.e. whether a certain feature is present at a given location in the image. If something moves in the input image, its activation will also move by the same amount in the output.

When performing a convolution, there are several aspects to define (Dumoulin and Visin, 2016). First of all, we have to define the **dimensions of a filter**. It usually has three dimensions *width*, *height* and *depth*. While the *depth* dimension matches the *depth* of the input image, the *width* and *height* are consideribly smaller than the input *width* and *height*. We frequently use square filters for these dimensions. Other shapes are also possible when we want to emphasize a particular

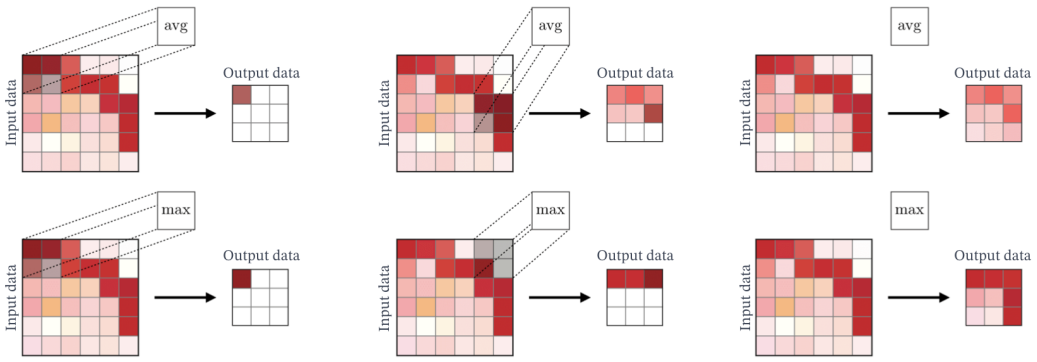


Figure 2.6: Two pooling strategies. **Max pooling**: we select the maximum value of the filter. **Mean pooling**: we average the values of the filter. Figure copyright CS 230 Deep learning.

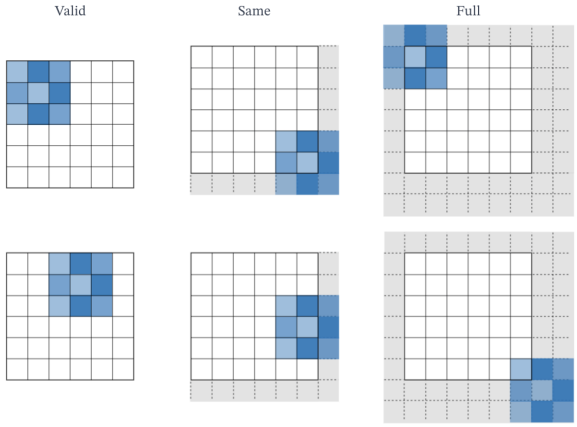


Figure 2.7: Three common padding strategies. The images show two positions of the same filters are in blue. The grey area indicates the zero-padding added for each case. **Valid**: No padding, we drop the last convolution if after applying the stride a part of the filter is 'outside' the input image. **Same**: padding such that output feature map has the same dimension that the input image. **Full**: maximum padding such that the last convolutions on each axis are applied on the limits of the input. Figure copyright CS 230 Deep learning.

dimension. **Stride** denotes the number of pixels on each axis[1] by which the filter moves after each operation. Strides bigger than one have less overlapped information and downsample the output. **Zero-padding** concatenates zeros to each side of input boundaries. This is done to obtain outputs with the same or higher dimension of the input (see Figure 2.7). Zero-padding is essential to perform a *'transposed convolution'* operation (Zeiler et al., 2010). Transposed convolutions are used when we want to change the order of the dimensions, e.i. having a bigger output than input. The output shape of a convolutional layer is defined by these parameters. It is usual to apply a special filter called **pooling** that does a final downsampling operation after a convolution operation. Pooling

[1]Since the *depth* dimension matches the *depth* of the input we slide only in the *width* and *height* axes

reduces the size of our array while keeping the most important features. It also produces spatial invariance and makes the features robust against noise and distortion. Pooling downsamples each *depth* independently, reducing only the height and width. The most used pooling strategies are *max* and *average* where the maximum and average value is, respectively, taken (see Figure 2.6). The digital image processing domain has developed many handcrafted filters such as the Laplacian filter for highlighting regions of rapid intensity change (edge detection).

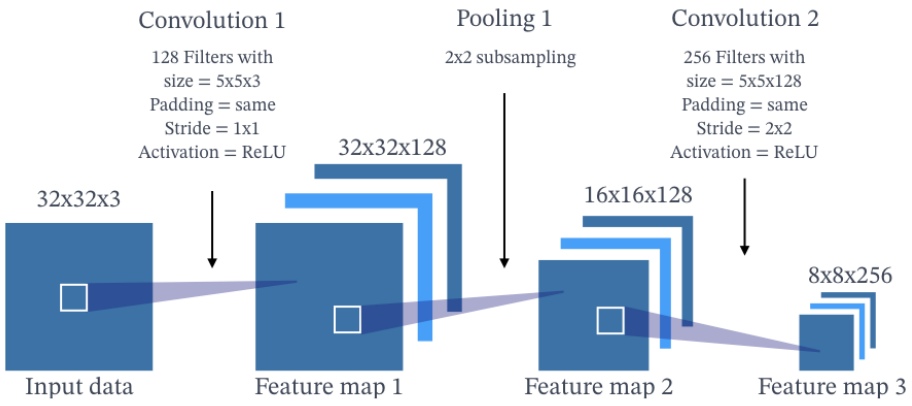


Figure 2.8: CNNs are composed of a set of convolutional layers. Each layer computes a convolution operation, a non-linearity transformation and most of the time also a downsampling phase. Each filter at a given convolution is slid over the input producing a feature map that stores filter activations. Feature maps are arranged along the depth dimension.

Summing up, filters are sparse (with only a few elements we can transform the whole input), robust to spatial transformations and they reuse parameters (the same filter are applied to multiple locations). This is ideal for dealing with images and efficiently reduce the number of parameters of DNN architectures. A CNN is, in essence, a set of convolutional layers, each one composed by a convolution, a non-linearity operation and a downsampling phase (see Figure 2.8). The non-linearity operation is applied after we slide the filter over the input. We can downsample either by using a stride (the downsampling is computed directly in the main convolution itself) or applying pooling operation after the non-linearity operation. At a given layer, we apply many different convolutions in parallel, each one with a different filter. The main characteristic of CNN architectures is that filters are not hand-designed but learned as part of the training process using the backpropagation algorithm. Hence, the values of a filter are learnable weights that are trained for detecting the important features without any human supervision, playing the role of a feature extractor. Each convolution transforms the input into a tensor of filter activations arranged along the depth dimension called "feature maps". We then apply the non-linearity. In a CNN, each convolutional layer learns filters of increasing complexity. Adding many layers increase the abstraction capacity of the net (see Figure 2.9). The first convolution layer extracts low-level visual features like oriented edges,

lines, end-points, and corners. The middle layers learn filters that detect parts of objects. The last layers have higher representations: they learn to recognize full objects, in different shapes and positions. CNNs learn such complex features by building on top of each convolutional layer.

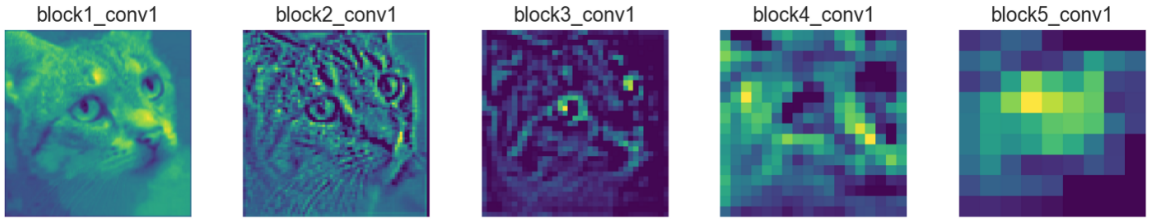


Figure 2.9: Visualization of one feature map per layer obtained using VGG16 (Simonyan and Zisserman, 2014). The first convolution layers (block1_conv1 and block2_conv1) compute feature maps that retain most of the original information, detecting low-level visual features like edges. As we go deeper in the network, the feature maps are more abstract but the original image is still visable (block3_conv1). It is interesting to see how important aspects such as the eyes and noise are more active. These layers focus on the classes in the image and less in the image itself. The deepest convolutional layers (block5_conv1) produce sparser feature maps, meaning the filters detect complex elements that may be not presented in every image. Figure copyright Applied Deep Learning

When defining a CNN architecture, there are many design choices such as the number of layers, filter sizes, the number of filters, stride, padding or non-linearity. This is task-depending and conventions are constantly updated. After the convolution layers, it is common to add several fully-connected layers to find patterns in the obtained high-level features. For that, we flatten the tensor into a 1D vector. This becomes quite standard for classification problems where the last fully-connected layer represents all the possible classes. CNN architectures are trained also with backpropagation and gradient descent.

CNN models are the most popular deep learning architecture. Complex architectures that stack multiple and different convolutional layers have revolutionized the digital image processing domain. They are also widely used in other domains such as recommender systems, speech recognition, natural language processing, or MIR. CNN architectures are the main tool we employ to develop our ideas in the next chapters.

2.1.4 Autoencoders

Autoencoders are DNN architectures that have as target value y the input x (Ballard, 1987). They compress the input into a lower-dimensional representation and reconstruct the output from it. The lower-dimensional representation (also called latent-space representation) serves as a compact "summary/compression" of the input. In practice, it is an internal hidden layer that describes the input only by a few variables. Autoencoders have two components: the encoder and the decoder (see Figure 2.10). The encoder compresses the input into the latent-space $e(x) = l$. Alternatively, the decoder reconstructs the original input using the latent-space information $d(l) = x'$ only. In fact,

autoencoders aim to learn an approximation to the identity function $d(e(x)) = x' \approx x$. Rather than a direct identity function, they add constraints for learning useful properties of the data. Commonly, the decoder architecture is the mirror image of the encoder but this is not mandatory. The only requirement is that the input and output must have the same dimensions so that the "loss function" $\mathcal{L}$ can directly compare point-wise them.

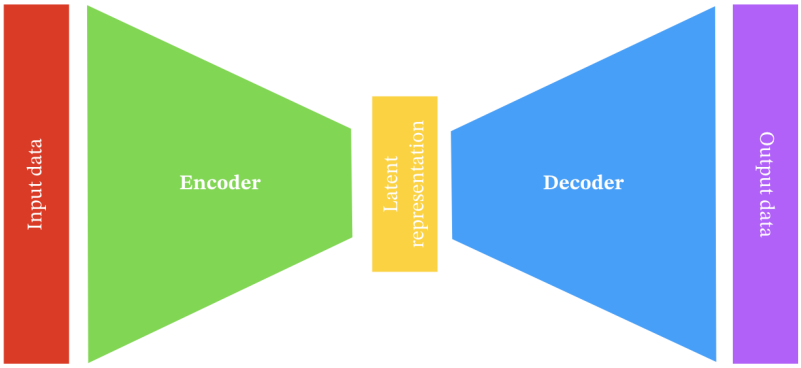


Figure 2.10: First the input passes through the encoder to project the data into the latent-space. Then the decoder produces the output only using the latent-space data. We train the system minimizing the differences between the input and output.

Autoencoders are one of the most popular unsupervised learning architectures. They belong to the self-supervised family because they generate their own labels from the training data. Autoencoders were originally used as compression techniques with losses (the final output is a close but degraded representation of the original). Soon, they were applied as robust denoising methods (Lu et al., 2013) by simply adding noise to the inputs and using as target the original noise-free data. They are also used as feature learning by removing the decoder and adding new layers for performing a particular task. This is usually combined with transfer learning, i.e. transferring the learned variables of an architecture to another architecture. In this case, the encoder must have the same architecture as the target dedicated net (the final net that performs the task). Once the autoencoder is trained, we use the weights of the encoder to initialize the weights of the target dedicated net (Masci et al., 2011; Zhuang et al., 2015). This helps overcoming the problem of insufficient label data in a supervised learning task. Modern autoencoders use stochastic mappings $p_{encoder}(h|x)$ and $p_{decoder}(x|h)$ for generative modeling, i.e. being able to generate new samples from the learned distribution. The most well-known example is Variational Autoencoder (Kingma and Welling, 2014)

2.1.5 U-Net

Inspired by autoencoders, the U-Net architecture (Ronneberger et al., 2015) has also an encoder/decoder mirror architecture based on CNN (see Figure 2.11). Each convolutional block in the encoder halves the size of the input and doubles the number of channels. The decoder obtains

the original size of the input by a stack of transposed convolutional operation. Both encoder and decoder have the same number of blocks. This architecture adds residual/skip connections (see Section 2.1.6) between layers at the same hierarchical level in the encoder and decoder, i.e. the input of each decoder block is both the output of the previous block and the output of the corresponding encoder layer. This ensures that the encoded features are directly used in the reconstruction. The output of the U-Net is not the input but a modification that highlights or isolates a particular aspect or a specific target location. Notice how this formalization is no longer unsupervised learning but rather supervised learning because it requires labeled data. The U-Net architecture is one of the main tools we employ in this thesis.

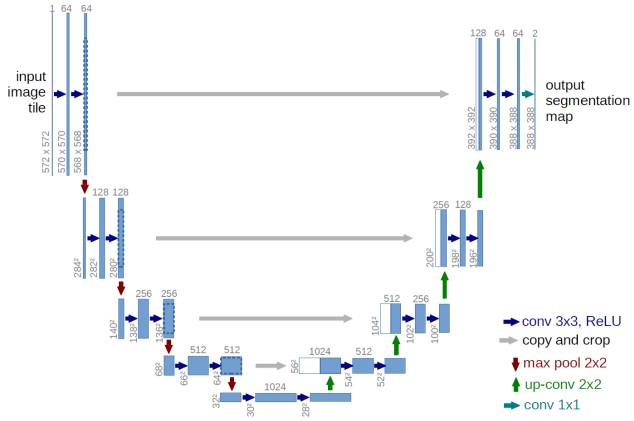


Figure 2.11: Original U-Net architecture. It follows a encoder decoder schema with residual/skip connections between layers at the same hierarchical level. The output is not the original input but rather a modification that highlights or isolated a particular aspect of the input. Figure copyright (Ronneberger et al., 2015)

2.1.6 Additional deep neural network components

In this section, we detail a set of common techniques used to speed-up the training, avoid overfitting and create more robust neural networks. These techniques are employed in our models.

Dropout

Dropout prevents over-fitting during training time. At each training iteration, we "drop" a random selection of a fixed number of the units (Srivastava and Salakhutdinov, 2014). Dropped neurons are disabled, not participating in the training process. Dropped neurons at one step are usually active at the next step. Using dropout prevents neurons co-adaptation (i.e. to be dependent on a small number of previous neurons). We explicitly force every neuron to be able to operate independently by learning robust features useful in conjunction with several random subsets of neurons. We avoid

dropout in the output layer (it is where we want to specialize each neuron to do something concrete). Dropout is not applied during inference.

Batch Normalization

Batch normalization improves the performance and stability of neural networks (Ioffe and Szegedy, 2015). We usually standardize (zero mean and standard deviation of one) the input data so that each feature has the same contribution, reducing the sensitivity to small changes. Batch normalization extends this idea by normalizing/standardizing activations in intermediate layers. At each iteration of the training process and given a minibatch, we normalize the output of one layer before applying the activation function. The normalization is done using the mean and standard deviation of the values in the current batch. We then feed it into the following layer. Batch normalization also adds two learnable parameters: a shift factor γ and scale factor β. These parameters restore the representation power of the network to take advantage of the non-linearity function in the case it cannot learn with that zero-mean and unit-variance constraint. They also control the needed mean and the variance of the layer which helps our optimization algorithm. In inference, we usually use an average of the accumulated mean and variance during training.

Batch normalization reduces the amount by what the hidden unit values shift around, giving the same importance at each input feature. It optimizes the training because networks learn faster (converge quickly), allows higher learning rates, reduces the sensitivity to the initial starting weights and keeps a controllable range of values avoiding saturations for some non-linearity activations (Goodfellow et al., 2016).

Data Augmentation

Overfitting happens because we have too few examples to train on. As a result, our model finds an overcomplex function that does not generalize. In the hypothetical case of having access to all the instances of the unknown joint probability distribution $p(x, y)$, we would not overfit because we would see every possible instance. Nevertheless, we only have access to $\mathcal{S}$. Data augmentation artificially enriches or "augments" the training set $\mathcal{S}$ by generating new instances. We aim to generate realistic $\mathcal{S}$ instances. The transformations should be learnable by the model, and not being simple noise. Data augmentation can rapidly increase the size of our training set, reducing overfitting. It is only performed on the training data, we do not modify the validation or test set.

There are many data augmentation techniques. Traditional methods apply random transformations to the existing instances in $\mathcal{S}$. This technique is very effective for image classification task. Dedicated strategies designed for particular tasks also enhance the accuracy and generalization ability (Mauch and Ewert, 2013). New augmentation techniques explore how to learn augmentations that best improve the ability of the net to correctly perform a task. These methods achieve state-of-the-art results (Perez and Wang, 2017; Cubuk et al., 2019).

Residual/skip connections

A residual/skip connection "connects" the output of one layer with the input of an earlier layer (He et al., 2016) (see Figure 2.12). These connections can skip multiple layers. Adding more layers increases the complexity and expressiveness of the network but also makes them much more expressive, difficult to train and adds more unpredictability. New layers define new independent functions. A new independent function does not guarantee increasing the expressive power of the network. This is only guaranteed when larger function classes contain the smaller ones (nested functions). This is the core idea behind residual/skip connection. Each additional layer should contain the identity function as one of its elements. Thereby, rather than parameterizing around a function $f(x)$ that outputs zero in its simplest form (weights are zeros), we parameterize around a function that outputs x (the identity) in its simplest form. With that each new layer deviate from the identity function, which still goes through the net. This leaves the outputs of the previous layers unchanged just that we could now do additional transformations. It also helps in a better gradient propagation. The residual/skip connection makes the gradient to pass unchanged to a previous connected layer, and also to the intermediate block to update its weights.

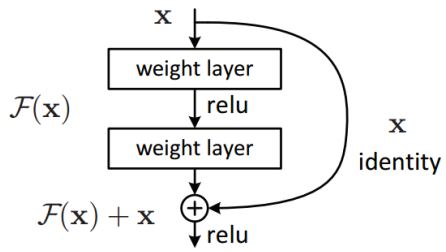


Figure 2.12: residual/skip connections connects the output of one layer with the input of another. The final output corresponds to the identity function in its simplest form. Figure copyright (He et al., 2016)

Residual/skip connections are implemented as summation or concatenation. Using element-wise summation can be seen as feature refinement through the various layers of the network. It is a compact solution that keeps the number of features fixed across blocks. On the other side, concatenating allows the subsequent layers to re-use middle representations, maintaining the original information. It has a better gradient propagation for deep architectures but it can lead to an exponential growth of the parameters.

2.2 Conclusion

In this chapter, we reviewed the collection of tools we use to develop our ideas. In the following chapters, we delve into the topics covered here explaining some aspects furthermore, adding some

more specific and advanced tools and showing how they have effectively applied for our tackled problems. We use them during the creation of our dataset and for exploring two MIR tasks: lyrics segmentation and source separation. When working with the audio signal, we employ spectral representations. Although they differ from traditional images in many aspects (e.g. the spatial correlation is much more complex and 'pixels' at a particular frequency do not only depend on their closer ones but also on 'pixels' at far frequencies, the harmonics), these spectral representations can be seen as images. Hence, we mostly employ CNN architectures.

Chapter 3

DALI: A Dataset of Audio with Lyric Information Aligned In Time

The central topic of this thesis is the multimodal analysis of singing voice investigating music and lyrics. Our research focuses on the vocals of a song. We are interested in the direct interaction between the audio signal and the lyrics. Thus, we need a specific kind of data: audio signals and their matched lyrics aligned in time. Nevertheless, there is a lack of large and good quality datasets of this kind. Lyrics aligned in time can be found for commercial purposes (LyricFind, Musixmatch or Music-story). Yet, they are private, not accessible outside host applications, come without audio, have only aligned text lines and do not contain vocal melody symbolic notation (notes). To carry out our research we need to have access to this kind of data. Thereby, the first contribution of this thesis is the Dataset of Aligned Lyric Information (DALI) (Meseguer-Brocal et al., 2018): a large dataset with time-aligned vocal notes and lyrics at four levels of granularity: notes (with their correspondent underlying phonemes), words, lines, and paragraphs. DALI has 5358 songs for the first version and 7756 for the second one.

In this chapter, we first define the dataset itself and discuss why DALI is needed. Then we explain the developed tools that come with it and analyze the information presented.

3.1 Motivation

Many MIR tasks are complex predictions estimated at a particular time instant such as note estimation or instrument recognition. To solve these tasks, researchers usually formulate their solutions in a supervised learning setting. In this paradigm, models use labeled data to discover functions that map input-output pairs (see Chapter 2). Having large, good quality and reality representative of the real world datasets is essential to success in any supervised learning problem. The generalization

of a model (i.e. to correctly determine the output of unseen input data) depends critically on the number of labeled examples (Sun et al., 2017).

The image processing domain has constantly improved thier models thanks to benchmark reference datasets such as MNIST (Le Cun et al., 1998), CIFAR (Krizhevsky, 2009), YouTube-8M (Abu-El-Haija et al., 2016) or ImageNet (Deng et al., 2009). These datasets are used, standardized and accessible by everybody enabling model comparisons.

Nevertheless, in MIR there is a lack of benchmark reference datasets. There are various reasons for this absence: legal problems, label complexity (each audio segment has fine time and/or frequency resolution), task diversity (the same audio excerpt can have many different labels related to the task at hand) and the need for expert knowledge (with possible disagreements). There are two main paths for creating datasets: either doing so manually or reusing/adapting existing resources. Although the former produces precise labels, it is time-consuming, and the resulting datasets are often rather small. Since existing resources with large data do not meet the MIR requirements, datasets that reuse/adapt them are usually noisy and biased.

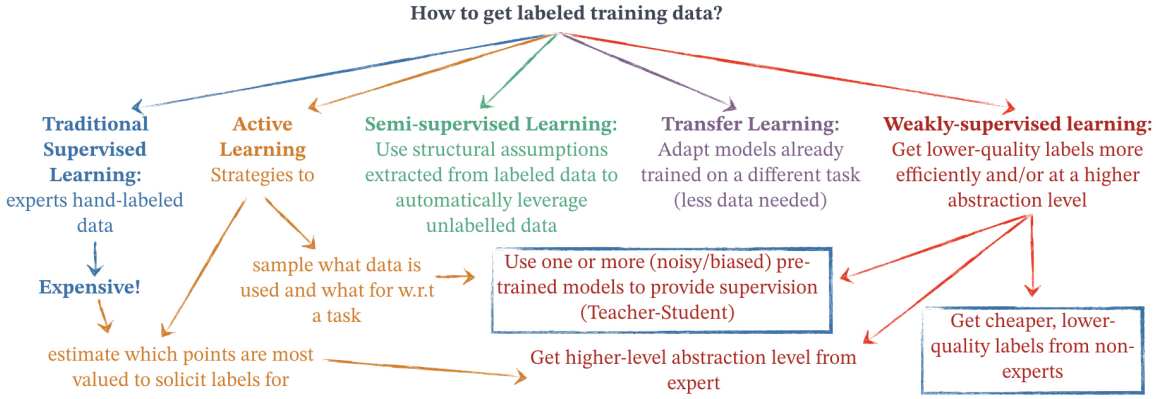


Figure 3.1: Schema of the different paradigms to deal with the problem of insufficient labeled data. DALI is framed inside Weakly supervised learning approches where cheap labels are obtained from non-experts. In Chapter 4 we will explore how to use pre-trained models to provide supervision for creating the DALI dataset.

Currently, many new areas have appeared to face the issue of insufficient labeled data (see Figure 3.1). **Semi-supervised learning** uses labeled data together with a large amount of unlabeled data (Zhu, 2005). Hereabouts, systems automatically leverage unlabeled data through deriving insights from the labeled one. **Active learning** estimates the most valuable points for which to solicit manual expert labels and explore strategies to automatically select what data is the most valuable to use and what for given a particular task (Settles, 2008; Krause et al., 2016). **Weakly supervised learning** (Mintz et al., 2009; Mnih and Hinton, 2012; Xiao et al., 2015) deals with low-quality labels (or at a higher abstraction level that needed, for instance having labels only at the audio excerpt and not the frame level) to infer the desired target information.

Many MIR datasets include *musical note events* labels, where a note event consists of a start time, end time and pitch. They are useful for a number of applications that bridge between the audio and symbolic domain, including symbolic music generation and melodic similarity. Instruments such as the piano produce relatively well-defined note events, where each key press defines the start of a note. Other instruments, such as the singing voice, produce more undefined note events, where the time boundaries are often related with changes in lyrics or simply as a function of our perception (Fürniss and Castellengo, 2016), and are therefore harder to annotate correctly. Reference datasets such as MedleyDB (Bittner et al., 2014) or MusicNet (Thickstun et al., 2017) with full audio tracks and musical note events labels and/or fundamental frequency have a great impact on the MIR research community.

Datasets providing note event annotations are created in a variety of ways.The most predominant MIR datasets are manually created, where notes are manually labeled by music experts, requiring the annotator to specify the start time, end time and pitch of every note event manually, aided by software such as Tony (Mauch et al., 2015). However, there are not many of such type and the final dataset is rather small as it consumes a large number of resources. Recently, large datasets created reusing and/or leveraging MIDI files from the Internet have been proposed (Meseguer-Brocal et al., 2018; Donahue et al., 2018; Raffel, 2016; Benzi et al., 2016; Fonseca et al., 2017; Donahue et al., 2018; Nieto et al., 2019; Maia et al., 2019; Yesiler et al., 2019). Yet, these sources are not designed for MIR needs, producing in noisy labels emerging the question of how unambiguous and accurate they are. Note data has also been collected automatically using instruments which "record" notes while being played, such as a Disklavier piano in the MAPS (Emiya et al., 2009) and MAESTRO (Hawthorne et al., 2019) datasets, or a hexaphonic guitar in the GuitarSet dataset (Xi et al., 2018). Data collected in this way is typically quite accurate, but may suffer from global alignment issues (Hawthorne et al., 2019) and can only be achieved for these special types of instruments. Another approach is to play a midi keyboard in time with a musical recording, and use the played midi events as the note annotations (Su and Yang, 2015) but this requires a highly skilled player in order to create accurate annotations.

In this thesis we are intesested in tasks related to singing voice which, despite being one of the most important elements in popular music, it is a lesser-studied topic in MIR community. Although many singing voice problems (e.g. singing voice detection or lyrics alignment) have been widely studied in the MIR community, singing voice was introduced as a standalone topic only a few years ago when it (Goto, 2014; Mesaros, 2013). This topic specially suffers from lack of benchmark dataset. Currently, researchers working in singing voice use small datasets. Each one is designed following different methodologies (Fujihara and Goto, 2012). Large datasets remain private (Humphrey et al., 2017; Stoller et al., 2019) or are vocal-only captures of amateur singers recorded on mobile phones that involve complex pre-prossessing (Smith, 2015; Kruspe, 2016; Gupta et al., 2018). This absence of reference datasets is a critical point that has been always neglected preventing the singing voice

community from training state-of-the-art machine learning algorithms and comparing their results.

Table 3.1: Comparison of the different datasets with lyrics aligned in time.

Dataset	**Number of songs**	**Language**	**Audio type**	**Granularity**
(Iskandar et al., 2006)	No training. 3 tests songs	English	Polyphonic	Syllables
(Wong et al., 2007)	14 songs divided into 70 segments with 20s long	Cantonese	Polyphonic	Words
(Müller et al., 2007)	100 songs	English	Polyphonic	Words
(Kan et al., 2008)	20 songs	English	Polyphonic	Section Lines
(Mesaros and Virtanen, 2010)	Training: 49 fragments ∼25 seconds for adapting a phonetic model Testing: 17 songs	English	Training: A Capella Testing: Vocals after source separation	Lines
(Hansen, 2012)	9 pop music songs	English	Both, Polyphonic A Capella	Words Lines
(Mauch et al., 2012)	20 pop music songs	English	Polyphonic	Words
DAMP dataset, (Smith, 2015)	34k **amateur** versions of 301 songs	English	Amateurs A Capella	Not time-aligned lyrics only textual lyrics
DAMPB dataset, (Kruspe, 2016)	A **DAMP** subset with 20 performances of 301 songs	English	Amateurs A Capella	Words Phonemes
(Dzhambazov, 2017)	70 fragments of 20 seconds	Chinese Turkish	Polyphonic	Phonemes
(Lee and Scott, 2017)	20 pop music songs	English	Polyphonic	Words
(Gupta et al., 2018)	A **DAMP** subset with 35662 segments of 10s long	English	Amateurs A Capella	Lines
$\text{Jamendo}_{\text{aligned}}$, (Ramona et al., 2008) (Stoller et al., 2019)	20 Creative commons songs	English	Polyphonic	Words
DALI v1 (Meseguer-Brocal et al., 2018)	5358 songs in full duration	Many	Polyphonic	Notes, words, lines and paragraphs
DALI v2	7756 songs in full duration	Many	Polyphonic	Notes, words, phonemes, lines and paragraphs

Table 3.1 contains an overview of public datasets with lyrics aligned in time. Most of these datasets are created in the context of lyrics alignment task. In this task, researchers try to assign start and end times to every fragment of textual information. Lyrics are inevitably language-dependent. Researchers have created several datasets for different languages: English (Kan et al., 2008; Iskandar et al., 2006; Gupta et al., 2018), Chinese (Wong et al., 2007; Dzhambazov, 2017), Turkish (Dzhambazov, 2017), German (Müller et al., 2007) and Japanese (Fujihara et al., 2011). Most datasets contain polyphonic popular music. There are many datasets with A Capella music (Kruspe, 2016). However, it is always difficult to migrate the methods to the polyphonic case (Mesaros and Virtanen, 2010). Datasets do not always contain the full duration audio track but often a shorter version (Gupta et al., 2018; Dzhambazov, 2017; Mesaros and Virtanen, 2010; Wong et al., 2007). If the tracks are complete, respective datasets are typically small. One of the goals of this thesis is to build a large and public dataset with audio, lyrics, and notes aligned in time, the DALI dataset.

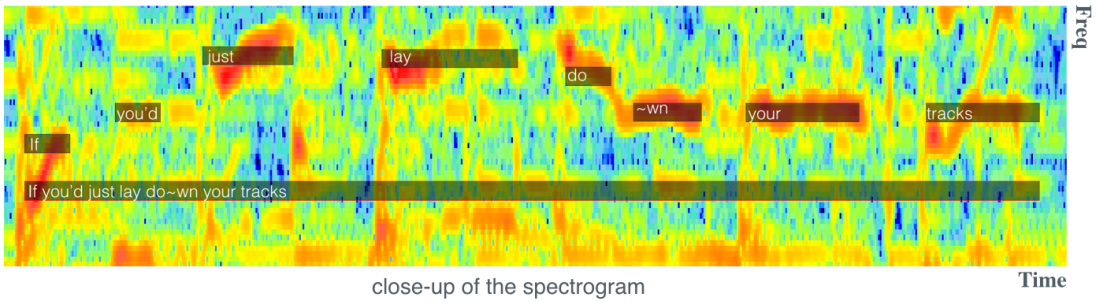


Figure 3.2: An example of the manual annotation overlap with its spectrogram. The close-up of the spectrogram illustrates the alignment for a small excerpt at two levels of granularity: notes and lines.

3.1.1 Our proposal

The DALI dataset: a large **D**ataset of synchronised **A**udio, **L**yrics and p**I**tch aims to serve as a reference dataset for the singing voice community. We presented it in 2018 at the International Society for Music Information Retrieval (ISMIR) conference (Meseguer-Brocal et al., 2018). It contains the audio of full songs each with – their audio in full-duration, – their time-aligned vocal melody and – their time-aligned lyrics. Thus, it contains *musical note events* and lyrics aligned information. Lyrics are described according to four levels of granularity: **notes** (the phonemes underlying a given note), **words**, **lines**, and **paragraphs**. It also provides additional metadata such as genre, language and musician and some multimodal information like album covers or links to video clips (see Figure 3.3).

The DALI dataset has not been created manually. Rather, we leveraged existing open-source karaoke resources where non-expert users manually annotated the lyrics and melody of a song.

Table 3.2: DALI dataset general overview

V	Songs	Artists	Genres	Languages	Decades
1.0	5358	2274	61	30	10
2.0	7756	2866	63	32	10
Multitracks	512	247	32	1	7

From these resources, we developed a system that finds the corresponding audio tracks and aligns the annotations to it. Our approach consists of constant interaction between dataset creation and learning models where they benefit from each other (this is described in detail in Chapter 4). DALI has 5358 songs for the first version and 7756 for the second one. There are also 512 songs in multitrack version (M) with two stems (**vocals** and **accompaniment**) and the final **mix** (see Table 3.2).

Figure 3.3: Example of metadata annotations.

3.2 Definition

The DALI dataset is a collection of songs described as a sequence of time-aligned lyrics, each one linked to its audio in full-duration. Annotations define a direct relationship between the audio and the lyrics represented as text information at different hierarchical levels. This is very useful for a wide variety of MIR problems such as lyrics alignment and transcription, melody extraction, structure analysis, hierarchical interaction or vocal source separation.

Time-aligned lyrics are described at four levels of granularity: **notes**, **words**, **lines** and **paragraphs**, from the deepest to the highest. The lyrics are described as a sequence of characters for all levels. For the multitrack and the second version, the **word** level also contains the lyrics as sequence of **phonemes**. Lyrics at the **note** level correspond to the syllable (or group of syllables) sung, and the frequency defines the musical notes for the vocal melody. The different granularity levels are vertically connected, i.e. one level is associated with its upper and lower levels. For instance, we know the words of a particular line, or which paragraph a line belongs to. In Figure 3.2, we illustrate an example with two levels of granularity: a line and its corresponding notes.

Table 3.3: Statistics for the different DALI dataset versions. One song can have several genres.

V	Average songs per artist	Average duration per song	Full duration	Top 3 genres	Top 3 languages	Top 3 decades
1.0	2.36	Audio: 231.95s With vocals: 118.87s	Audio: 344.9hrs With vocals: 176.9hrs	Pop: 2662 Rock: 2079 Alternative: 869	ENG: 4018 GER: 434 FRA: 213	00s: 2318 90s: 1020 10s: 668
2.0	2.71	Audio: 226.78s With vocals: 114.73s	Audio: 488.1hrs With vocals: 247.2hrs	Pop: 3726 Rock: 2794 Alternative: 1241	ENG: 5913 GER: 615 FRA: 304	2000s: 3248 1990s: 1409 2010s: 1153
M	2.07	Audio: 220.83s With vocals: 98.97s	Audio: 35.4hrs With vocals: 14.1hrs	Rock: 312 Pop: 258 Alternative: 162	ENG: 512	2000s: 188 1990s: 103 1980s: 93

3.2.1 Formal definition

In DALI, songs are defined as:

$$S = \{A_{notes}, A_{words}, A_{lines}, A_{paragraphs}\} \tag{3.1}$$

where each granularity level g with K elements is a sequence of aligned segments:

$$A_g = (a_{k,g})_{k=1}^{K_g} \text{ where } a_{k,g} = (t_k^0, t_k^1, f_k, l_k, i_k)_g \tag{3.2}$$

with t_k^0 and t_k^1 being a text segment's start and end times (in seconds) with $t_k^0 < t_k^1$, f_k a tuple (f_{min}, f_{max}) with the frequency range (in Hz) covered by all the notes in the segment (at the note level $f_{min} = f_{max}$, a vocal note), l_k the actual lyric's information and $i_k = j$ the index that links an annotation $a_{k,g}$ with its corresponding upper granularity level annotation $a_{j,g+1}$. The text segment's events for a song are ordered and non-overlapping - that is, $t_k^1 \leq t_{k+1}^0 \forall k$. Note how the annotations define a unique connection in time between the **musical** and **textual** domains.

3.3 Dataset analysis

DALI has 5358 songs for its first version (Meseguer-Brocal et al., 2018), 7756 for the second one and 512 for the multitrack subset. This means a total of 344.9, 488.1 and 35.4 hours of music respectively with 176.9, 247.2 and 14.1 hours with vocals. In terms of annotations, there are more than 3.6 and 8.7 million $a_{k,g}$ for version 1 and 2 and 486k for the multitrack. The average $a_{k,g}$ per song is 679, 710 and 950. There are, on average, 2.36 (2.71) songs per artist and 119s (115s) with vocal durations per song, in v1 (v2) (see Table 3.3).

As seen in Table 3.2, DALI has a great range of artists, genres, languages and decades. Most of the songs are from popular genres like Pop or Rock and the 2000s. The most predominant language is English but there are also many songs in German and French.

Table 3.4: Proposed split with respect to the time correlation values described in Chapter 4.4

	Correlations	Tracks
Test	$NCC_t >= .94$	1.0: 167 2.0: 402
Validation	$.94 > NCC_t >= .925$	1.0: 423 2.0: 439
Train	$.925 > NCC_t >= .8$	1.0: 4768 2.0: 6915

Finally, using the correlation scores described at Chapter 4.4, we propose to split DALI into 3 sets: train, validation, and test (see Table 3.4). However, depending on the task at hand (e.g. analyzing only English songs or lyrics alignment) other splits are possible.

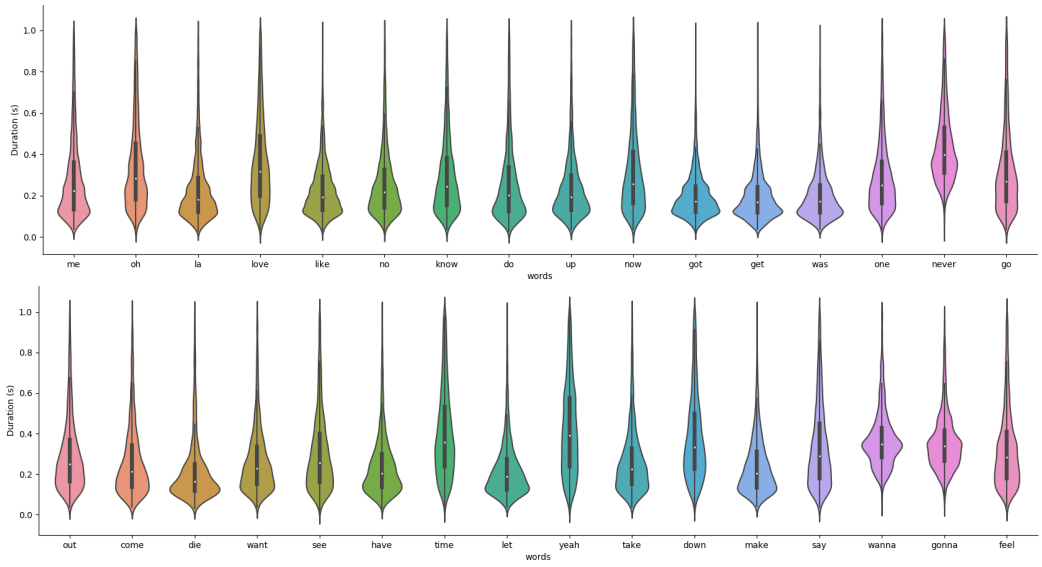


Figure 3.4: Distribution of the duration in seconds of the most common 32 words (after removing pronouns and articles) for the second version of DALI.

3.4 Working with DALI

3.4.1 Tools

The richness of DALI renders the data complex. Therefore, it would be difficult to use in a raw format such as `JSON` or `XML`. To overcome this limitation, we have developed a specific `Python` package that has all the necessary tools to access the dataset. It can be found at `https://github.com/gabolsgabs/DALI` (see Figure 3.5) and easily be installed using pip[1].

[1] `https://pypi.org/project/DALI-dataset/`

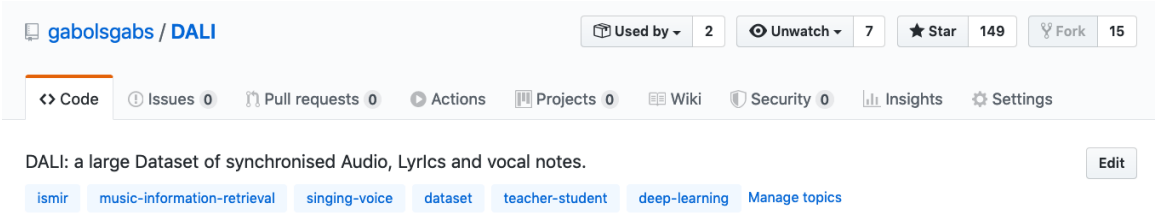


Figure 3.5: The DALI package is available at `https://github.com/gabolsgabs/DALI` and contains all the necessary tools for working with the annotations.

A song is represented as the Python class, `Annotations` (see Figure 3.6). `Annotations` instances have two attributes `info` and `annotations`. The attribute `info` contains the metadata, the scores that guide the quality of the annotations (see Chapter 4.4 and Chapter 5.1.2) and links to the audio.

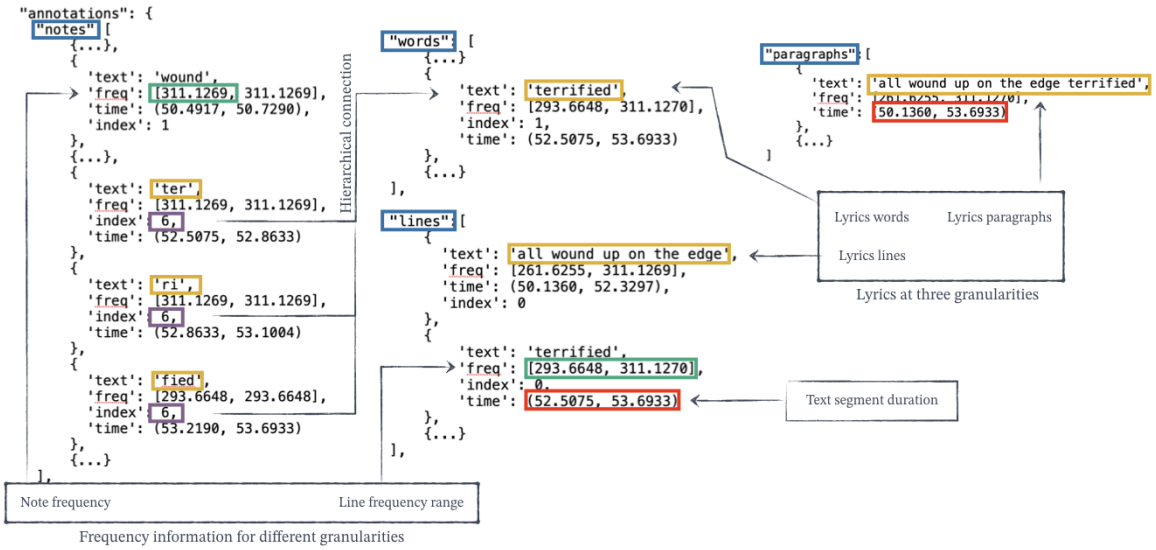


Figure 3.6: Annotations class data example with four levels of granularity. Note how the word 'terrified' is sung in three different notes. Thanks to the index each level of granularity is connected with its upper one.

The attribute `annotations` contains the aligned segments $a_{k,g}$. We can work in two modes *`horizontal`* and *`vertical`*, and easily change from one to the other. The *`horizontal`* mode stores the granularity levels in isolation, providing access to all its segments. The *`vertical`* mode connects levels vertically across the hierarchy. A segment at a given granularity contains all its deeper segments e.g. a line has links to all its words and notes, allowing the study of hierarchical relationships.

The package also includes a group of additional tools. There are general tools for reading the whole dataset and automatically retrieving the audio from the internet. We also provide tools for

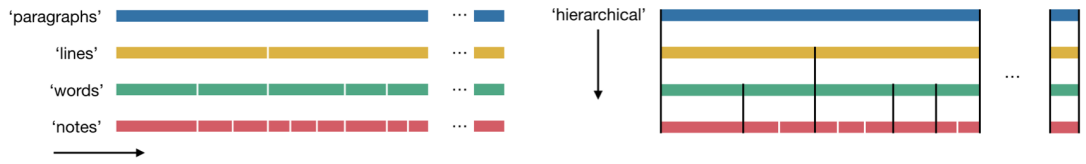


Figure 3.7: Annotations are presented in two different modes. On the left, the *horizontal* mode that stores the granularity levels in isolation. On the right, the *vertical* mode that connects all levels hierarchically.

working with individual granularity levels i.e. transforming the data into vectors or matrices with a given time resolution, to manually correct the global parameters or to re-compute the alignment for different audio than the original one (see Chapter 4.4). For a detail explanation of how to use DALI, we refer to the tutorial at `https://github.com/gabolsgabs/DALI`.

3.4.2 Distribution

Figure 3.8: The actual DALI annotations are available at `https://zenodo.org/record/2577915` and can be downloaded after agreeing to use them only for research.

Each DALI dataset version is presented as a set of gzip files. Each file encloses an instance of the class `Annotations`. The different versions can be found at `https://zenodo.org/record/2577915` (see Figure 3.8). They are distributed as open-source under an `Academic Free License, AFL` [2]. Each version is described following the MIR corpora description (Peeters and Fort, 2012) and has a fingerprint (a `MD5` checksum file (Rivest, 1992)) that verifies the integrity of it.

Finally, the DALI dataset is also part of mirdata (Bittner et al., 2019), which provides a standard framework for MIR datasets as well as a fingerprint (also a `MD5` file) per `Annotations` instance and audio track that verifies their integrity.

3.4.3 Reproducibility

One of the main problems in DALI is the restriction on sharing the audio of each song. This complicates the comparison of the results or may end up in misaligned annotations if different audio is used. We suggest three ways to overcome this issue:

[2] https://opensource.org/licenses/AFL-3.0

1. to use the tools provided to retrieve the audio we use directly from YouTube. Unfortunately, some of the links may be broken and not all the audio might be available.

2. to use a different audio version and reproduce the alignment techniques as in Chapter 4.4. We provide all the tools for the task and grant a model (second generation (see Chapter 4.5.4)) for computing the singing voice activation vector needed.

3. to send us the computation needed to be run on our audio. The user has to agree to distribute the new feature to other users (at `zenodo`) as the main melody representation (f_0) computed in Chapter 5.1.2.

Finally, the multitrack version is not distributed.

3.5 Conclusions

In this chapter we detailed our first contribution, the DALI dataset. We formally defined it and performed an statistical analysis of its content to get to know better its peculiarities. We have also introduced the developed tools that help us to work with this complex data. We use DALI for tackling our research problems in the following chapters.

Chapter 4

Creating DALI: a real case scenario[1]

Creating the DALI dataset represents a challenging research question. We start with songs manually annotated by non-expert users into notes and lyrics of the vocal melody. These annotations come without audio and they are only described by artist name and song title. Also, the annotations are not always accurate enough to be used as a MIR dataset. To create a clean dataset, we need to 1) find the corresponding audio used and 2) improve the quality of the annotations' time information.

For each annotated songs, we retrieve a set of audio candidates from YouTube. Each one is turned into a Singing Voice Probability vector (SVP) over time using a Singing Voice Detector (SVD), based on a deep CNN architecture. We find the best candidate and correct the annotations' time information by comparing this SVP to the annotated Voice Annotation Sequence (VAS), derived from the time-aligned lyrics. The quality of this matching is restricted by the performances of the SVD system. Whereas our original model retrieves good annotations, it does not align properly the annotations to it. To improve the SVP, we adopt a teacher-student paradigm (Hinton et al., 2014; Li et al., 2014). In this paradigm, a first model named the teacher is trained using a clean and well-annotated controlled dataset. The teacher is then used to select the best-aligned tracks defining a new training set (annotation/audio matches). This set is used to train a new SVD system, the student. Using the "knowledge" learned by the teacher on clean data, the student has proved to perform better than the teacher on the SVD task.

Our method is well motivated by Active learning and Weakly-supervised Learning (see Figure 3.1). It establishes a loop whereby dataset creation and model learning interact, benefiting each other by progressively improving our model using the collected data. At the same time, we correct and enhance the data every time we update the model. This process creates an improved DALI

[1]Some of the work reported here was done in collaboration with Alice Cohen-Hadria who implemented the Singing Voice detection model and trained the base line versions.

after each iteration.

4.1 From karaoke annotations to structured MIR data

DALI stands as a solution to the absence of large reference dataset with lyrics and vocal notes time-aligned. These types of annotations are hard to obtain and very time-consuming to create. We opt for reusing/adapting existing resources. Concretely, our solution is to look outside the field of MIR. We turn our attention to karaoke video games where users sing along with the music. They win points singing accurately which is measured by comparing the sung melody with time and frequency-aligned references. Therefore, large datasets of time-aligned melodic data and lyrics exist. Apart from commercial karaoke games, there are several active and large karaoke open-data communities. In those, non-expert users exchange text files that contain the reference annotations without any professional revision. We retrieved 13,339 of these karaoke annotation files. However, they need to be adapted to the requirements of MIR applications[2].

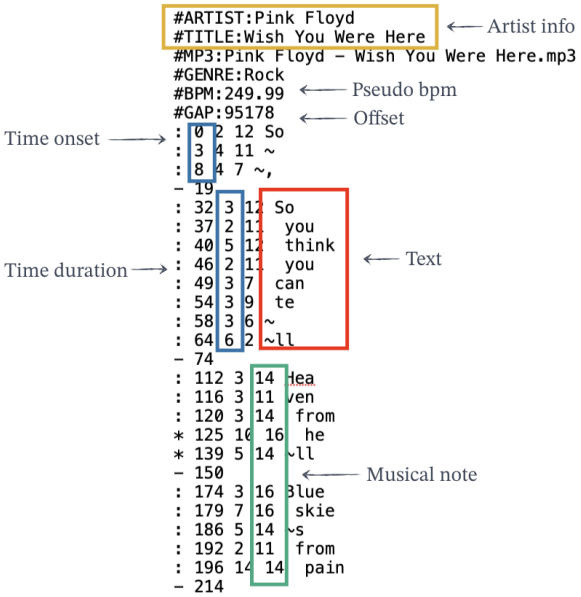


Figure 4.1: Example of the raw data contained in a karaoke file

As illustrated in Figure 4.1, karaoke users create and exchange annotations in text files. Each file contains::

- the `song_title` and `artist_name`.
- a sequence of triplets {`time, musical-note, text`} with annotations,

[2]Standardized format with a time in seconds, frequency in hertz and the normalized annotations with characters in the utf8 format

- the `offset_time` (start of the sequence) and the `frame rate` (annotation time-grid),

We first transform the raw information into useful data obtaining the time in seconds and the frequency note. Then, we create the different levels of granularity: **notes** (and textual information underlying a given note), **words**, **lines** and **paragraphs**. The note, word, and line levels are encoded in the retrieved files. We deduct the paragraph level as follows.

The paragraph level. Using the `song_title` and `artist_name`, we connect each raw annotation file to the Web Audio Semantic Aggregated in the Browser for Indexation (WASABI) (Meseguer-Brocal et al., 2017) dataset, a semantic database of song metadata collected from various music databases. WASABI provides lyrics grouped in lines and paragraphs, in a text-only form. We created the paragraph-level by merging the two representations (melodic note-based annotations from karaoke annotations and text-only annotations from WASABI) in a text to text alignment. Let l^m be our existing raw *lines* and p^m the *paragraph* we want to obtain. Similarly, p^t represents the *target* paragraph in WASABI and l^t its *lines*. Our task is to progressively merge a set of l^m such that the new p^m is maximally similar to an existing p^t. This is not trivial. l^m and l^t differ in some regards: l^m tends to be shorter, some lines might be missing in one domain, and p^t can be rearranged/scrambled. An example of the merging system is shown in Figure 4.2.

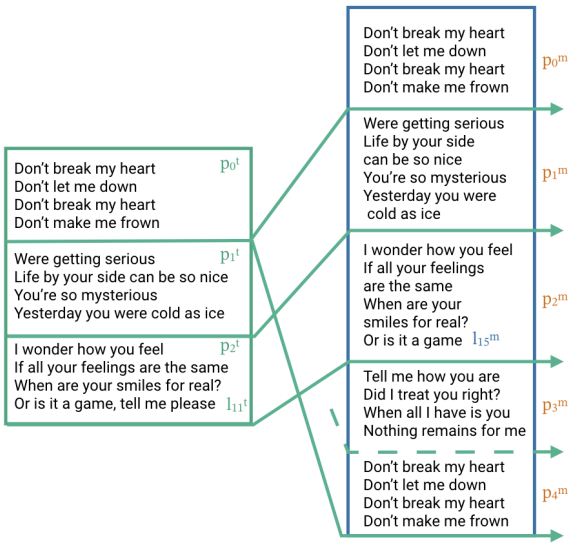


Figure 4.2: [Left] Target lyrics lines and paragraphs as provided in WASABI [Right] The melody paragraphs p^m are created by merging the melody lines l^m into an existing target paragraph p^t. Note how line l^t_{11} in p^t_2 has no counterpart in l^m_* and chorus p^m_3 does not appear in any p^t_*.

The phoneme information. The phonetic information is computed only for the word level. We use the Grapheme-to-Phoneme (G2P) [3] system by CMU Sphinx[4] at Carnegie Mellon University. This

[3] https://github.com/cmusphinx/g2p-seq2seq
[4] https://cmusphinx.github.io/wiki/

Table 4.1: Overview of terms: definition of each term used in this chapter. NCC_t is defined at Section 4.4.

Term	Definition
`Notes`	time-aligned symbolic vocal melody annotations.
`Annotation`	basic alignment unit as a tuple of: time (start and duration in frames), musical note (with 0 = C3) and text.
`A file with annotations`	group of annotations that define the alignment of a particular song.
`Offset_time` (o)	the start of the annotations.
`Frame rate` (fr)	the reciprocal of the annotation grid size.
`Voice annotation sequence` ($vas(t) \in \{0, 1\}$)	a vector that defines when the singing voice (SV) is active according to the karaoke-users' annotations.
`Predictions` ($\hat{p}(t) \in [0, 1]$)	probability sequence whether or not singing voice is active at any frame provided by our singing voice detection.
`Labels`	label sequence of well-known ground truth datasets checked by the MIR community.
`Teacher`	SV detection (SVD) system used for selecting audio candidates and aligning the annotations to them.
`Student`	new SVD system trained on $vas(t)$ of the subset selected by the Teacher after $NCC(\hat{o}, \hat{fr}) \geq T_{corr}$.

model uses a `tensor2tensor` transformer architecture that relies on global dependencies between input and output. The final phoneme level has the text information transcribed into a vocabulary of 39 different phoneme symbols defined in the Carnegie Mellon Pronouncing Dictionary (CMUdict)[5]. The phoneme information is only available for multitrack and version two.

The metadata. Additionally, WASABI provides extra multi-modal information such as cover images, links to video clips, metadata, biography, and expert comments.

4.2 Finding the correct audio

The annotations are now ready to be used. Nevertheless, they come without audio. Linking each annotation file with its proper audio track is a mandatory step to perform any audio-related task. The same `song_title` and `artist_name` may have many different versions (studio, radio, edit, live or remix) and each one can have a different lyrics alignment. Hence, we need to find the correct version used by the karaoke-users to do the annotations. The WASABI metadata provides exactly all the versions for a pair `song_title` and `artist_name`. With this information, we query YouTube to recover a collection of audio candidates. This is similar to other works where authors used chroma features and diagonal matching to align jazz solos and audio candidates from YouTube (Balke et al., 2018).

To find the correct audio used by the karaoke-users we need to answer to three questions:

1. *Is the correct audio among the candidates?*

2. *If there are more than one, which is the best?*

[5]https://github.com/cmusphinx/cmudict

3. *Do annotations need to be adapted to the final audio to perfectly match it?*

Moreover, the fact that users are amateurs may lead to errors and some annotations have to be discarded. This introduces a fourth question: ***are annotations good enough to be used?***.

To answer these questions, we need to measure the accuracy of an audio candidate to an aligned annotation. Therefore, we need to find a common representation for both audio and text.

4.2.1 Working with Audio and Lyrics part 1: Annotations as audio

In this section, we review our first attempts to find the correct audio candidates. These attempts were guided by the idea that the annotations can be seen as "audio features" i.e. we focus on how to transform them to fit audio representation spaces.

Lyrics-alignment.

We first explored lyrics alignment techniques (Fujihara and Goto, 2012; Kruspe, 2016; Dzhambazov, 2017) which aims at automatically synchronizing sung lyrics with their written versions. In other words, these techniques aim at determinating **where** lyrics appeared in the audio. From a more technical point of view, it is the problem of finding the correct temporal location in the audio of limited textual units.

The problem starts with a given audio signal and its corresponding lyrics. The goal is to assign start and end times to every fragment of textual information. These fragments can have different levels of granularity: phonemes, syllables, words, phrases and paragraph and the difficulty of the tasks increases with granular levels.

Most of the approaches work with polyphonic popular music. Methods developed for A Capella music (Kruspe, 2016) are always difficult to migrate to the polyphonic music (Mesaros and Virtanen, 2010). Lyrics are inevitably language-dependent. They have been several studies for English (Kan et al., 2008; Iskandar et al., 2006), Chinese (Wong et al., 2007; Dzhambazov, 2017), Turkish (Dzhambazov, 2017), German (Müller et al., 2007) and Japanese (Fujihara et al., 2011). In theory, these systems can be adapted to any language but no experiments have been conducted to prove this.

There are several dimensions over which methods can be classified, being **features** employed one of them. Most of the approaches use phonetic features following a speech recognition paradigm. That is, for each phoneme an acoustics model is created which aims to capture the traits of a specific phoneme. Due to the lack of annotated and isolated data, there has been a lot of effort in adapting speech models to the particularities of singing voice (Mesaros and Virtanen, 2010) or particular singers (Fujihara et al., 2011). There are few approaches that do not use phonetic features: methods that align the fundamental frequency (F0) of the singing voice with the tone of each word (only for tonal languages such as Cantonese) (Wong et al., 2007) or use the phoneme duration with prior structures (detected by chord and rhythm) (Kan et al., 2008).

Studies can be also grouped along the **alignment** technique used. There are two approaches: forced or non-forced. **Forced** alignment **aligns all** the textual elements with audio segments. It is a method inherited from the speech community and it is the most widely used. Usually, the whole lyrics is expanded to a network of phonetic models (including a silence element). Each model yields a likelihood according to an input features vector. Thus, the audio track is synchronized with the lyric net forcing all elements in the net to have a connection with an audio segment. Most of the forced methods use techniques such as Hidden Markov Models (HMMs) or Viterbi algorithm. Recently, more sophisticated methods such as Dynamic Bayesian Networks (DBN) have been proposed for dealing not only with the transitions between phonemes but also with the complementary context (Dzhambazov, 2017). In contrast, **non-forced** systems have the freedom to **not align all** the textual unit with audio extracts.

Finally, there is a set of studies that use the complementary context around lyrics for improving the performance. Elements such as structure (Iskandar et al., 2006; Lee and Cremer, 2008), melodic phrases and metric circles (Dzhambazov, 2017), chord progressions (Mauch et al., 2011), MIDI files (Müller et al., 2007) or manually-annotated segmentation labels (such as Chorus and Verse) (Lee and Cremer, 2008) are used as cue for alignment.

When we first tackled this problem the recent techniques based on DNN did not exist. Some authors proposed to use Automatic Speech Recognition (ASR) imperfect transcription to align lyrics and A Capella singing signals, in a semi-supervised way (Gupta et al., 2018). The authors transfered a method conceived for A Capella audio to the polyphonic case. They adopted a neural network acoustic model trained on a large number of solo singing vocals by using its weights as initialization for the training with polyphonic music (Gupta et al., 2019). New methods also use connectionist temporal classification (CTC) loss to train a model that extracts a character probability matrix from the audio signal. This matrix is used to align the target lyrics to it (Stoller et al., 2019).

The problem with these techniques is that they are complex and have in the phonetic model its main limitation (there is no phonetic model trained on singing voice rather adaptations from speech). Having a good dataset such as DALI is a key element for developing good lyrics alignment algorithms. They also assume that the pair audio-lyrics are correct which is not our case. Finally, with these techniques it is difficult to know if the resulting lyrics are well aligned. Hence, we discarded this direction.

Score-alignment.

Annotations can be transformed into a sequence of musical notes. This allows us to formalize the problem as a score alignment approaches (Cont et al., 2007; Soulez et al., 2003; Raffel, 2016). Score alignment techniques are similar to the previous ones but they focus on the harmonic distribution of the spectrogram instead of the phonetic traits.

The problem with these techniques is that they assume that every event in the audio has a

representation in the musical score. This is not our case because there is a lack of information for the non-vocal areas (we only have notes for the singing voice). Silence in the score does not match with silence in the audio signal. Furthermore, the musical background is not represented whatsoever in our score sequences. They also presume that the musical score is correct which again is not our case. Thus, these systems do not solve any of our issues and result in misaligned sequences.

Dominant melody estimation.

A natural solution to this issue is to assume that the dominant melody in the audio corresponds only to the singing voice, and then align both signals. We investigate this idea using Melodia (Salamon and Gómez, 2012), since it was one of the state-of-the-art dominant melody detection algorithms[6]. Nevertheless, after several experiments, we found that the estimated melody is not sufficiently precise and does not always correspond to the vocal melody. Hence, both sequences can not be properly aligned.

Accordingly, we did not persist in this direction. However, we will explore some of these ideas in future works, especially in the scenario in which we already have the annotations globally well alligned with the audio. In this scenario, annotated notes are a key element to solve local alignment problems.

4.2.2 Working with Audio and Lyrics part 2: Audio as annotations

Instead of focusing on how to transform the annotations to fit the audio representation spaces, our solution is to focus on how to transform the audio into the annotation space. We do that by converting the audio into an SVP over time $\hat{p}(t)$:

$$\hat{p}(t) = \begin{cases} 1, & \text{if there is singing voice at time frame } t \\ 0, & \text{if there is no singing voice at time frame } t \end{cases}$$

We denote by $\hat{p}(t)$ the **predictions**. The model in charge of doing so is an SVD system and it is described in next section. Likewise, we can transform the lyrics annotations into $vas(t)$ into a VAS with value 1 when there is vocal annotations and 0 otherwise:

$$vas(t) = \begin{cases} 1, & \text{if there is singing voice} \\ 0, & \text{otherwise} \end{cases}$$

We only focus on the vocal segments contained in A_g (see Chapter 3.2), discarding the frequency and text information:

$$A_g = (a_{k,g})_{k=1}^{K_g} \text{ where } a_{k,g} = (t_k^0, t_k^1)_g \tag{4.1}$$

[6]At the moment these experiments were carried out, new DNN architectures such as (Bittner et al., 2017) did not exist.

This type of problem is typically not estimated at the segment observation level, but rather at the *frame* level. The annotations of a given granularity level A_g are divided into an evenly spaced time grid $(r_i = H \cdot i)_{i=0}^{m}$ where H is a constant defining the spacing between time stamps ($H = 14$ ms to have the same time resolution as $\hat{p}(t)$). Although we can construct $vas(t)$ at any level of granularity, we only work at the note level because it is the finest.

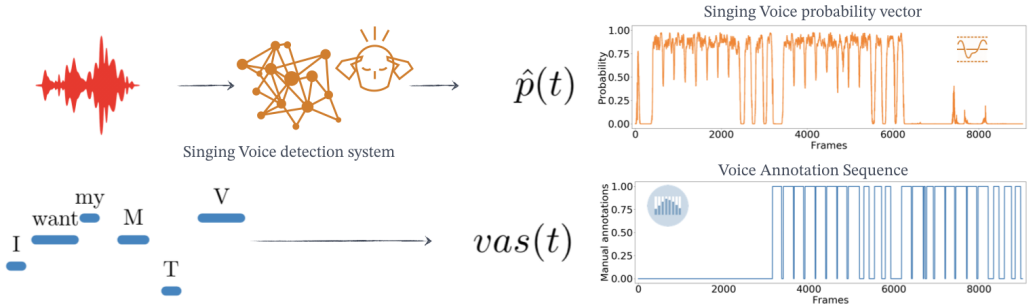


Figure 4.3: Illustration of the computation of $\hat{p}(t)$ and $vas(t)$

These two vectors can be directly compared (see Figure 4.3). Our hypothesis is that with a highly accurate SVD system and exact annotations, both vectors should be identical. Consequently, it should be reasonably easy to use them to find the correct audio.

4.3 The Singing Voice detection problem

The problem of transforming the audio into SVP over time is called the *singing voice detection* task. In contrast with the *alignment* tasks, in the *singing voice detection* we aim to know, from the audio signal analysis, the probability of having a singing voice or not.

4.3.1 Previous approaches

Most approaches for the *singing voice detection* task share a common architecture. Short-time observations are used to train a classifier that discriminates observations (per frame) in vocal or non-vocal classes. The final stream of predictions is then post-processed to reduce artifacts.

Classification approach.

Early works explore classification techniques such as Support Vector Machines (SVMs) or K-Nearest Neighbors (KNN) based on different audio descriptors (Rocamora and Herrera, 2007). In (Ramona et al., 2008), the authors use as audio features the centroid, width, asymmetry, slope, decreasing, flux and similar temporal statistical moments, along with their first and second derivatives and 13-order Mel Frequency Cepstral Coefficient (MFCC) resulting in a raw feature vector of 116 components, at each time frame. After this extraction process, an SVM is used to classify each frame into a singing or non-singing class. The same work also presents the Jamendo dataset, used in

future experiments. On this dataset, the authors report a 71.8% accuracy without post-processing and 82.2% accuracy with a Hidden Markov Model (HMM) post-processing. Similarly, MFCC features extracted from a predominant melody extraction, like the pitch fluctuation feature and MFCC of Re-Synthesized Predominant Voice are also used (Mauch et al., 2011). As in the previous method, classification is performed using SVMs applied to the frame level vector of features. They test their model on 100 songs from the RWC Music Dataset (Goto et al., 2002) reporting accuracy of 87.2%.

Using specific voice properties. Other approaches also try to use specific vocal traits (Regnier and Peeters, 2009). Here, the particularities of vocal vibrato and tremolo (average rate, average extent, and presence of both modulations) are exploited to discriminate instrumental and singing voice segments. This approach achieves a recall of the singing voice class of 83.57% on the Jamendo dataset. Some authors adopt speech recognition systems for the particularities of singing voice (Berenzweig and Ellis, 2001). Given a time frame, they create a feature vector, composed of 13 perceptual linear predictions (PLP) coefficients. The idea is to exploit the resemblance between singing voice and spoken voice when compared with non-vocal music segments. From the PLP coefficients, the authors extract different features. They use a dataset composed of 246 15-second fragments recorded at random from FM radio in 1996. The authors report a 73,9% accuracy on the singing / non-singing classification task.

Singing voice detection as a preprocessing step. In (Berenzweig et al., 2002), Multi-Layer Perceptron (MLP) on each feature vector (13 perceptual linear prediction (PLP) coefficients) was employed to distinguish between singing and non-singing segments. To provide context, 5 consecutive frames are presented to the network. This segmentation in vocal and non-vocal segments is then used to perform an artist classification, the target task of this study. Similarly, to perform audio lyrics alignment, authors proposed a vocal activity detection method based on an HMM on top of a Gaussian mixture model (GMM) which models the power of harmonic components (Fujihara et al., 2011). They then compare the given harmonic structure with only those in the dataset that have similar F0 values. This first step helps at isolating the vocal segment, to reconstruct them as a separate audio track and then perform the lyrics alignment. Lyrics transcription (Mesaros, 2013) or source separation (Simpson et al., 2015) trained then to obtain ideal binary masks also use the *singing voice detection* as a pre-processing-step.

Deep learning. Over the past few years, works have focused on the use of DNN techniques. Some researchers propose the use of Recurrent Neural Networks (RNN) (Lehner et al., 2015) on a 60-dimensional feature vector (30-dimensional MFCC and their first order derivative) along with the cross-correlation of each filter-bank spectrum of a time frame (called a fluctogram), spectral contraction and spectral flatness. They reported 89.42% accuracy on the Jamendo dataset. Bidirectional Long Short-Term Memory (BLSTM), a class of recurrent neural network, were also used in (Leglaive et al., 2015). Authors chose to represent the audio signal as a combination of 1) a harmonic part and 2) a percussive part, using a double stage HPSS as proposed in (Tachibana et al., 2010). Each

part is then transformed into a Mel-spectrogram and passes to the BLSTM network. They report a 91,5% accuracy on the Jamendo dataset. A comprehensive discussion of these approaches can be found at (Lee et al., 2018a).

Finally, the use of CNN is also popular. They were introduced in combination with data augmentation to increase the size of the training set (Schlüter and Grill, 2015) or trained on weakly labeled data (each 30 seconds excerpt is labeled as containing singing voice or not, but not at the frame of the excerpt) along with a three steps training strategy (Schlüter, 2016). A Constant-Q input model trained on a very large private dataset (mined from Spotify resources) obtains an accuracy of 87,8% on the Jamendo dataset (Humphrey et al., 2017).

4.3.2 Our model

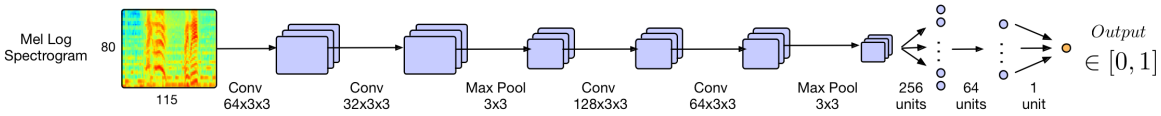


Figure 4.4: Architecture of our Singing Voice Detection (SVD) system using CNN proposed by (Schlüter and Grill, 2015).

The model we use is based on the CNN proposed by (Schlüter and Grill, 2015). It follows a standard CNN architecture with two fully-connected layers and a final neuron that provides $\hat{p}(t)$ for the center time-frame of the patch. The input is a sequence of patches of 80 Log-amplitued Mel bands coefficients over 115 consecutive time frames (0.014 seconds per frame). MLS applies a bank of triangular filters to the power of the Discrete Fourier transform (DFT) spectral coefficients (see Figure 5.7). These filters are designed based on the Mel-scale to be more discriminative at lower frequencies and less at higher frequencies. This mimics the nonlinear critical bands of the human ear. We then compute the logarithm of the magnitude in each new bin. Figure 4.4 shows the architecture of the network. The model is trained on binary target using a binary cross-entropy loss-function, ADAMAX optimizer, mini-batch of 128, and 10 epochs. We chose this architecture because it has proved to be easy to implement, fast and more important, robust for the singing voice detection task.

4.4 Normalized cross-correlation

At this stage, audio and annotation are described as vectors over time $\hat{p}(t) \in [0, 1]$ and $vas_{o,fr}(t) \in \{0, 1\}$.(see Figure 4.3). We measure their similarity using the Normalized Cross-Correlation (NCC) . This similarity is not only important in recovering the right audio and finding the best alignment, but also in filtering imprecise annotations. (see Section 4.2).

We measure the similarity between $\hat{p}(t) \in [0,1]$ and $vas_{o,fr}(t) \in \{0,1\}$ using the NCC, which is the normalized version (between 0 and 1) of the cross-correlation. The cross-correlation measures the similarity between two digital sequences by sliding one -y- over the other -x-. We use it when we search a shorter sequence in a longer sequence or when there is a relative displacement between x and y (our case). At each value of the lag l, we calculate the correlation between them i.e. their degree of similarity. This results in a new function that describes where y best matches with x. The highest correlation coefficient represents the best fits position l between the two sequences. This is the position that interests us.

$$R_{x,y}(l) = \sum_{l=-\infty}^{\infty} x(t)y(t-l) \tag{4.2}$$

Since we are interested in global alignment we found this technique more precise than others such as Dynamic Time Warping (DTW). Indeed, DTW finds the minimal cost path for the alignment of two complete sequences. To do so, it can locally warps the annotations which usually deforms them rather than correcting them. It is also costly to compute and its score is not directly normalized, which prevents us from selecting the right candidate.

The $vas(t)$ depends on the parameters `offset_time` (o) and the `frame rate` (fr), $vas_{o,fr}(t)$. Hence, the alignment between $\hat{p}$ and vas depends on their correctness. While o defines the beginning of the annotations, fr controls the time grid size. When changing fr, the grid size is modified by a constant value that compresses or stretches the annotations as a whole respecting the global structure. Our NCC formula is as follows:

$$NCC(o,fr) = \frac{\sum_t vas_{o,fr}(t-o)\hat{p}(t)}{\sqrt{\sum_t vas_{o,fr}(t-o)^2}\sqrt{\sum_t \hat{p}(t)^2}}. \tag{4.3}$$

For a particular fr value, $NCC(o,fr)$ can be used to estimate the best $\hat{o}$ to align both sequences. We obtain the optimal $\hat{fr}$ using a brute force search in an interval α[7] of values around fr:

$$(\hat{fr},\hat{o}) = \underset{fr\in[fr-\alpha,fr+\alpha],\ o}{\arg\max} NCC(o,fr). \tag{4.4}$$

This automatically obtains the $\hat{o}$ and $\hat{fr}$ values that best align $\hat{p}(t)$ and $vas_{o,fr}(t)$ and yields a similarity value between 0 and 1.

This similarity automatically obtains the $\hat{o}$ and $\hat{fr}$ values that best align $\hat{p}(t)$ and $vas_{o,fr}(t)$. $NCC(o,fr)$ scores between 0 and 1. We compute the $NCC(\hat{o},\hat{fr})$ for all the audio candidates of a given $vas_{o,fr}$. Using this score we can both find the best candidate (highest score) and establish if the best candidate is good enough to be kept. To this end, we fix a threshold $NCC(\hat{o},\hat{fr}) \geq T_{corr}$ to keep only the accurate matches and discarding those for which the correct audio could not be found, and those with insufficiently accurate annotations (see Figure 4.5).

[7] we use $\alpha = fr * 0.05$

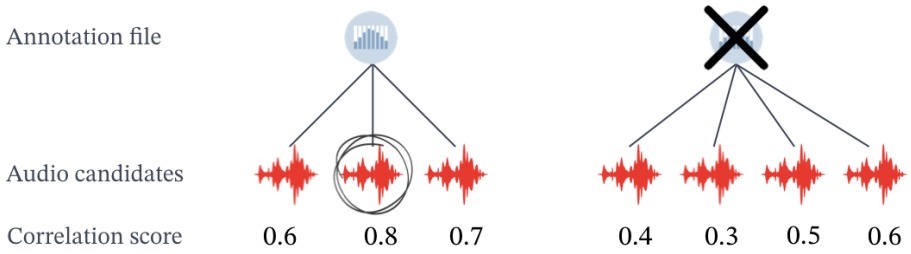


Figure 4.5: In order to discover its best alignment for each candidate, annotations are modified by changing its `offset_time` and `frame rate`. The candidate with the highest NCC value is our final audio track. This similarity is not only important in recovering the correct audio and finding the best alignment but also in determining if annotations are accurate enough. Fixing a threshold we can filter imprecise annotations.

The value of T_{corr} has been empirically found to be $T_{corr} = 0.8$. We have set up a high threshold to ensure that a good proportion of our chosen audio and labels are quite well annotated. This strategy is similar to the ones used in active learning, where, instead of labeling and using all possible data, we find ways of selecting the accurate data. The whole process is summarized in Figure 4.6.

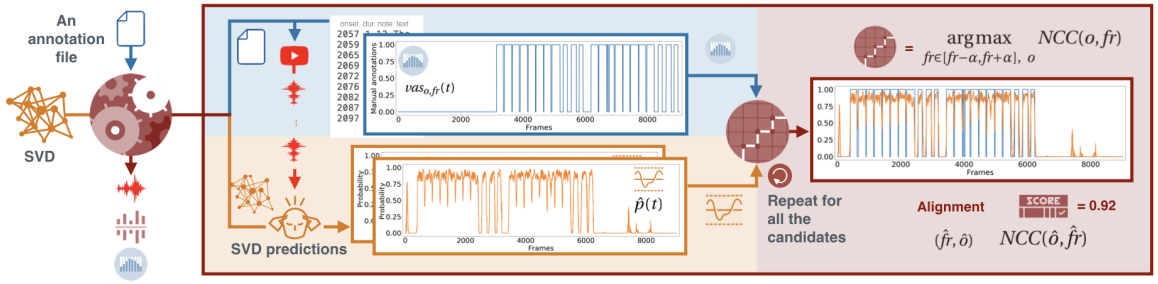


Figure 4.6: The input is an $vas_{o,fr}(t)$ (blue part) and a set of audio candidates retrieved from Youtube. The similarity estimation method uses a SVD model (orange part) to convert each candidate in a $\hat{p}(t)$ (orange part). We measure the similarity between the $vas_{o,fr}(t)$ and each $\hat{p}(t)$ using the cross-correlation method $\arg\max_{fr,o} NCC(o, fr)$ (garnet part). The output is the audio file with the highest $NCC(\hat{o}, \hat{fr})$ and the annotations aligned to it using $\hat{fr}$ and $\hat{o}$.

At this point and after manually examining the obtained alignments, we noticed that the quality of the process strongly depends on the quality of $\hat{p}(t)$. The $\hat{p}(t)$ obtained with the baseline SVD systems is sufficient to correctly identify the audio (although false negatives still exist), but not to align the annotations. Thus, we need to improve $\hat{p}(t)$. With increasing $\hat{p}(t)$, we will find more suitable matches and align the annotations more precisely (more accurate $\hat{o}$ and $\hat{fr}$).

There are two possibilities to do so: to develop a novel SVD system or to train[8] the existing

[8]We train each new model from scratch, not using transfer learning.

architecture with better data. Since DALI is considerably larger (around 2000) than similar datasets (around 100), we choose the latter. This idea re-uses all of the labeled data created in the previous step to train a better SVD system.

4.5 Improving DALI: The teacher student paradigm

In this section we show how we can take advantage of the data we just retrieved and aligned using a teacher-student paradigm.

4.5.1 Previous work

The two main agents of this paradigm are: the *'teacher'* and the *'student'*. The *teacher* is trained with labels of well-known ground truth datasets (often manually annotated) and used to label some unlabeled data. The new labels given by the teacher are used for training the *student(s)*. *Student* indirectly acquires the desired knowledge by mimicking the "*teacher's* behavior". This paradigm was originally introduced as a model compression technique to transfer knowledge from larger architectures to smaller ones ((Bucilua et al., 2006)). Small models (the students) are trained on a larger dataset labeled by large models (the teachers). A more general formalization of this knowledge distillation trains the student in both the teachers' labels and the training data (Hinton et al., 2014). In the context of Deep Learning, the teacher can also automatically remove layers in the architecture of the student, automating the compression process (Ashok et al., 2017).

The teacher-student paradigm is also used as a solution to overcome the problem of insufficient labeled training data, for instance for speech recognition (Watanabe et al., 2017; Wu and Lerch, 2017) or and multilingual models (Cui et al., 2017). Since manual labeling is a time-consuming task, the teacher-student paradigm explores the use of unlabeled data for supervised problems. The teachers (trained on labeled datasets automatically) label unlabeled data on a (usually) larger dataset used for training the students. This way of applying the teacher-student paradigm is one most popular methods employed in **semi-supervised** learning. This paradigm is still relatively undeveloped in MIR. One of the few examples that applies it to automatic drum transcription is (Wu and Lerch, 2017), in which the teacher labels the student dataset of drum recordings.

All of these works report that the students improve over the performance of the teachers. Therefore, this learning paradigm meets our requirements.

4.5.2 Our Teacher student

Our goal is to improve our SVD system. We use the teacher to select the retrieved audio and align the annotation to it. This new data is used for training a new SVD system. Our hypothesis is that if $\hat{p}(t)$ becomes better, the $\arg\max\ NCC(o, fr)$ will find better matches and align more precisely

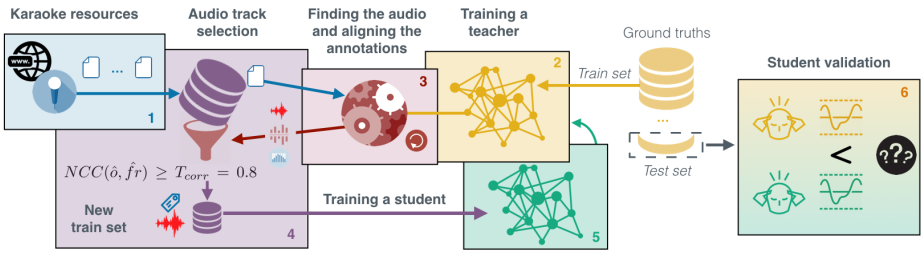


Figure 4.7: The DALI dataset creation using the teacher-student paradigm.

the annotations to the audio (more accurate $\hat{o}$ and $\hat{fr}$). As a result, we obtain a better DALI dataset. This larger dataset can then be used to train a new SVD system which again, can be used to find more and better matches improving and increasing the DALI dataset. This can be repeated iteratively. After our first iteration and using our best SVD system, we reach 5358 songs. We then perform a second iteration that defines the current 7756 songs.

Our teachers do not label directly the input training data of the students but rather select the audio and align the annotations to it (see Section 4.3). Similar to noisy label strategies such as active learning or weakly learning, we use a threshold to separate good and bad data points. But, instead of doing this dynamically during training, we filter it statically once an SVD model is trained. This process is summarized in Figure 4.7 and detailed in (Meseguer-Brocal et al., 2018):

1- Blue. The retrieved karaoke annotation files are converted to an annotation voice sequence $vas(t)$.

2- Yellow. We train a SVD, the **teacher**, either using clean ground-truth datasets or after the first iteration using DALI annotations (green arrow).

3- Garnet. With the teacher $\hat{p}(t)$ and the $vas(t)$ we compute the NCC to find the best audio candidate and alignment parameters $\hat{fr}, \hat{o}$ (see Section 4.4).

4- Purple. We select the pair audio-annotation with $NCC(\hat{o}, \hat{fr}) \geq T_{corr} = 0.8$. This set defines a new training set (and a DALI version).

5- Green. Using the new data we train a new SVD, called the **student**[9].

6- Yellow-Green. The two systems, teacher and student, are compared on the clean ground-truth test set to check that the new system performs indeed better than the previous one.

To train a new a **student** (step 5) we need to define the true value we want to model (target values p to be minimized in the loss $\mathcal{L}(p, \hat{p})$ of the SVD model). There are three choices. We can either use:

a) the predicted value $\hat{p}$ as provided by the **teacher** (this is the usual teacher-student paradigm).

[9]We retrained from scratch, **not** adapting the previous models.

b) the value of the *vas* corresponding to the annotations after aligning them using $\hat{fr}$ and $\hat{o}$.

c) a combination of both: keeping only the frames for which $\hat{p}(t) = vas(t)$.

Up to now, and since *vas* has been found more precise than $\hat{p}$, we only investigated option **b)**. This differs from other works where they use as target the output of the teacher (option **a)**). In our approach, the teacher 'filters' and 'corrects' the source of knowledge from which the student has to learn. Metaphorically speaking, instead of telling him exactly which 'sentences', our teacher tells the student which 'books' to read.

This process incrementally adds more good audio-annotations pairs. We perform this three times as summarized in Figure 4.9. With this process, we are simultaneously improving the SVD model and the dataset. Besides, this is also an indirect way of examining the quality of annotations: a well-performing system trained only with this data show us that the time alignment of the annotation is correct enough for this task.

4.5.3 First generation

Ground-Truth datasets

We use three ground-truth datasets to train the first teachers: *Jamendo* (Ramona et al., 2008), *MedleyDB* (Bittner et al., 2014) and a third one that merges both *J+M*. They are accurately labeled but small. In Figure 4.8 we present the mean duration of Jamendo and MedleyDB. We can see that MedleyDB contains track with many durations.

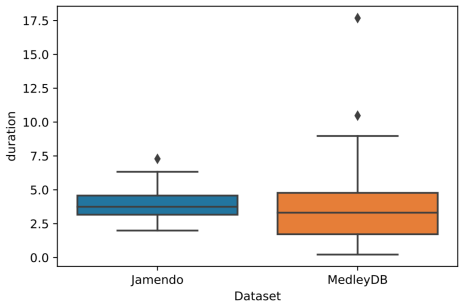


Figure 4.8: Box plot of duration of Jamendo and MedleyDB datasets datasets

Jamendo is a music website that offers free music streaming and download. Crawling this website, the authors downloaded 93 tracks with Creative Commons license (Ramona et al., 2008). Each track has been manually annotated by the same person. The dataset is split in a training set of 61 songs, 16 songs are kept for the validation set and 16 songs for the testing set.

MedleyDB (Bittner et al., 2014) contains 122 songs in multi-tracks, with mix, stems of different instruments. The dataset is annotated in melody F0 (for 108 tracks), instrument activations and genre (for all tracks). Among the 122 songs, 52 are instrumental only, 70 contains vocals. The

melody annotations were generated semi-automatically. A monophonic pitch tracking algorithm was used on the separated stems, the pYin algorithm. This gave the authors a good initial estimate of the F0 curve. For the automation of the instrument activations, a standard "envelope follower" technique on each stem was used[10]. After this automatic step, the annotations were refined by five human musician annotators.

Each dataset is split into a train, validation and test part using an artist filter[11]. We keep *Jamendo* and *MedleyDB* for testing the different SVD systems.

The Teachers

We train three teachers using the training part of each ground-truth set. The teachers select the audio and align the annotations as described in Section 4.4 creating three new datasets (DALI v0) with 2440, 2673 and 1596 items for the teacher *J+M*, *Jamendo* and *MedleyDB* respectively.

Table 4.2: Audio candidates intersection in percentage for filtering threshold = 80.

	J+M	*Jamendo*	*MedleyDB*	Three
J+M (2440)	100	91.4	61.6	58.8
Jamendo (2673)	83.4	100	55.2	53.6
MedleyDB (1596)	94.2	92.4	100	89.8

Table 4.2 presents the intersection of the three sets. It indicates how many tracks of each new set are also in the other two sets. For example, the 89.8% (bottom right) of the selected tracks using the *MedleyDB* teacher are also present within the ones selected by the *J+M* teacher or the *Jamendo* teacher. Also, the 91.4% (top second left) of the selected tracks using the *Jamendo* teacher are within the ones selected by the *J+M* teacher. This table shows that the three teachers agree most of the time on selecting the correct audio for a given annotation.

The Students

We train three different students. Among the possible target values: $\hat{p}$ given by the **teacher** -as common in the teacher-student paradigm-, *vas* after being aligned using *NCC* or a combination of both; we use the *vas*. We have found this vector to be more accurate than $\hat{p}$. In our approach, the teacher 'filters' and 'corrects' the source of knowledge from which the student learns. Each student is trained with different data since each teacher may find different audio-annotations pairs and different alignments (each one gets a different $\hat{p}$ which leads to different $\hat{fr}, \hat{o}$ values).

We hypothesize that if we have a more accurate $\hat{p}$, we can create a better DALI. In Table 4.3 and 4.4, we observe how the students outperform the teachers in both the singing voice detection task and the alignment experiment. Thus, they produce better $\hat{p}$. Furthermore, we assume that if

[10]It consists of half-wave rectification, compression, smoothing and down-sampling.
[11]No artist who appears in the training set can appear in the test set.

we use these SVD systems, we will retrieve better audio and have a more accurate alignment. For this reason we use the **student** based on ***J+M*** that obtains the best results to create DALI version one with 5358 songs(Meseguer-Brocal et al., 2018).

4.5.4 Second generation

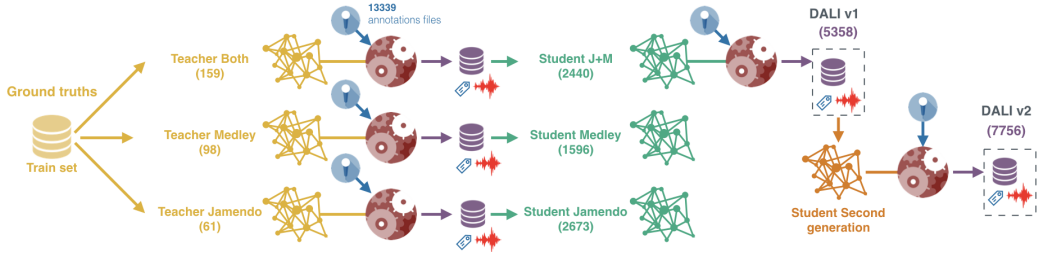


Figure 4.9: We create three SVD systems (*teachers*) using the ground truth dataset (Jamendo, MedleyDB and Both). The three systems generate three new datasets (DALI v0) used to train three new SVD systems (the *students* first generation). Now, we use the best student, *J+M*, to define DALI v1 released in (Meseguer-Brocal et al., 2018). DALI v1.0 is used to train a second-generation student that creates DALI v2 in (Meseguer-Brocal et al., 2020b)

We now use as a teacher the best student of the first iteration, the student ***J+M***. We again repeat the process described in Section 4.4. In this case, the teacher is not trained on any ground-truth but with DALI v1 and using the well aligned *vas* as a target. We split DALI v1 (5358 tracks) into three sets: 5253 for training, 100 for validation (the ones with higher NCC) and 105 for testing. The test set has been manually annotated (right $\hat{fr}$ and $\hat{o}$) and constitutes our ground-truth for future experiments[12].

The new SVD (student of the second generation) obtains even better results in both the singing voice detection task and the alignment experiment (see Section 4.6). Hence, we assume that another repetition of the process will output a better DALI. Indeed, this is the DALI v2 with 7756 audio-annotations pairs.

4.6 Experiments for validating the new Singing Voice Detection systems

We validate the performance of each model on two different tasks: the singing voice detection and the *vas* alignment result of the $\arg\max_{fr\in[fr-\alpha,fr+\alpha],\ o} NCC(o, fr)$. We demonstrate that generally, students perform much better than their teacher for both tasks.

[12]Note that this split is different from the proposed in Section 3.3 because of the nature of the experiments carried out

4.6.1 Singing voice detection

This task validates the accuracy of the predictions $\hat{p}(t)$ in the singing voice classification task. Results are indicated in Table 4.3 where e.g. '*S(T(J+M_Train))* (2673)' refers to the student trained on the 2673 audio-annotation pairs and the *vas* values obtained with the Teacher *T(J+M_Train)* trained on *J+M_Train* set.

We evaluate the performances of the various SVD models using the test and test+train (when possible) set for the *Jamendo* and *MedleyDB* ground-truths. We measure the frame accuracy for each model as followed. Firstly, we compute the threshold that defines which $\hat{p}(t)$ values are considered 1 (there is singing voice) or 0 (no singing voice) using the validation set of each ground-truth. The threshold might be different for each model and/or dataset. Then, we compute the binary accuracy score per track and finally the mean and standard deviation.

Table 4.3: Performances of ***singing voice*** detection, measured as mean accuracy and standard deviation, for the teachers and students. Number of tracks in brackets. Nomenclature: T = Teacher, S = Student, J = *Jamendo*, M = *MedleyDB*, *J+M* = *Jamendo* + *MedleyDB*, 2G = second generation, the name of the teacher used for training a student in bracket.

Test_sets / SVD system	*J_Test* (16)	*M_Test* (36)	***J_Test+Train*** (77)	***M_Test+Train*** (98)
T(J_Train) (61)	88.95% ± 5.71	83.27% ± 16.6	-	81.83% ± 16.8
S(T(J_Train)) (2673)	87.08% ± 6.75	82.05% ± 15.3	87.87% ± 6.34	84.00% ± 13.9
T(M_Train) (98)	76.61% ± 12.5	84.14% ± 17.4	76.32% ± 11.2	-
S(T(M_Train)) (1596)	82.73% ± 10.6	79.89% ± 17.8	84.12% ± 9.00	82.03% ± 16.4
T(J+M_Train) (159)	83.63% ± 7.13	83.24% ± 13.9	-	-
S(T(J+M_Train)) (2440)	87.79% ± 8.82	85.87% ± 13.6	89.09% ± 6.21	86.78% ± 12.3
2G(S(T(J+M_Train))) (5253)	**93.37% ± 3.61**	**88.64% ± 13.0**	**92.70% ± 3.85**	**88.90% ± 11.7**

Baseline SVD. We first test the teachers. *T(J_Train)* obtains the best results on *J_Test* (89%). *T(M_Train)* obtains the best results on *M_Test* (84%)[13] In both cases, since training and testing are performed on the two parts of the same dataset, they share similar audio characteristics. Therefore, these results are artificially high.

To best demonstrate the generalization of the trained SVD systems, we need to test them in a cross-dataset scenario, namely training and testing on different datasets. Comparing the performances on different test sets gives a sense of how good is the real generalization of the model. Indeed, in this scenario, the results are quite different. Applying *T(J_Train)* on *M_Test+Train* the results decrease to 82% (a 7% drop). Moreover, when applying *T(M_Train)* on *J_Test+Train* the results decrease to 76% (an 8% drop). Consequently, we can say that the teachers do not generalize very well.

Lastly, the *T(J+M_Train))* trained on J+M_train performs worse on both *J_Test* (84%) and *M_Test* (83%) than their non-joined teacher (89% and 84%). This result is surprising and remain

[13]There is a constant effect observed in all the experiments: a great result variability while testing on *MedleyDB*. We hypothesize that this is due to having a great number of instrumental songs in *MedleyDB* (62 out of 122).

unexplained. Unfortunately, we cannot prove its generalization since this system cannot be tested on *M_Test+Train* nor *J_Test+Train* because it has been trained with the training tracks.

First students. We now test the students. We hypothesize that students improve the results from the teachers due to the fact they have been trained using more data. Especially, we assume that their generalization to unseen data will be better. It is important to note that students are always evaluated in a cross-dataset scenario since their training set does not contain any track from *Jamendo* or *MedleyDB*. Hence, there are not artificially high results.

The student *S(T(J_Train))* based on *T(J_Train)* outperforms its teacher. When applied to *M_Test+Train*, it reaches 84% which is slightly higher than the performances of the *T(J_Train)* directly (82%). It also reaches similar results in the *J_Test+Train* 88% than its teacher in its own test set (89%).

Likewise, the student *S(T(M_Train))* based on *T(M_Train)* is also better than its teacher in the cross dataset scenario *J_Test+Train* with 84% while T(*M_Train*) (76%). This case is even better than the previous one since there is an 8% improvement. It also gets 82% on the *M_Test+Train* that is close to the 84% from its teacher in its own test set.

Finally, it is also true for the performances computed with the student *S(T(J+M_Train))* based on *T(J+M_Train)*. This system can be only compared with its teacher in the test dataset (not the test+train). Here, when applied either to *J_Test* or *M_Test*, it reaches 87% (88% on *Jamendo* and 86% on *MedleyDB*) which is above the *T(J+M_Train)* (83.5%). Also, this student performs as good (or above) as the other two previous teachers, *T(J_Train)* on *J_Test* (89%) and *T(M_Train)* on *M_Test* (84%). Besides, if we focus on the test+train section, this student outperforms any system on any dataset with 89% on *J_Test+Train* and 87% on *M_Test+Train*. These are very interesting results because this is the best student but it has not been trained with the best teacher (which is *Jamendo*). This SVD system is the one used for defining the first DALI dataset.

Second student. In this scenario, we hypothesize that the new student improves the results from previous students or teachers not only because it sees more data (5253) but also because this data is better than any previous one. To that end, we rely on the alignment results shown in Section 4.6.2, which demonstrates that the students produce also a better annotation alignment, therefore a more accurate target value while training.

In this second iteration, there is only one system: the *Second Generation 2G(S(T(J+M_Train)))* trained on the DALI dataset version 1 defined and aligned by the student *S(T(J+M_Train))*. The new SVD system confirms our hypothesis and outperforms notably any existing system. It gets 93% on *J_Test* and 89% on *M_Test* which are the best results on these test sets. These results are even higher than the artificially high ones obtained by the system trained directly on them where *T(J_Train)* gets 89% on *J_Test* and *T(M_Train)* 83% on *MedleyDB*. The *Second Generation* is also the one that generalizes the best. It reaches 93% on *Jamendo* _test+train and 89% on *MedleyDB* _test+train, which is more precise than our best previous system (the student based on *J+M_Train*)

that has 89% and 87% respectively.

This experiment proves that students work much better in the cross-dataset scenario (real generalization) when the train-set and test-set are from different datasets. It is important to note that the accuracy of the student networks is higher than that of the teachers, even if they have been trained on imperfect data.

4.6.2 On alignment

We hypothesize that a more accurate $\hat{p}$ leads to a better DALI. To prove this, we measure the precision of the o and fr computed by each SVD system. We have manually annotated 105 songs of DALI v1.0, i.e. finding the $\hat{fr}$ and $\hat{o}$ values that give the best global alignment. These songs are our ground-truth data. We measure how far the estimated $\hat{fr}$ and $\hat{o}$ diverge from the manually annotated ones. We name these deviation $offset_d$ and fr_d.

Table 4.4: ***Alignment*** performances for the teachers and students. Mean offset deviation in seconds, mean frame deviation in frames and pos = position in the classification.

	mean offset rank	pos	mean $offset_d$	mean fr rank	pos	mean fr_d
T(J_Train) (61)	2.79 ± .48	4	0.082 ± 0.17	1.18 ± .41	4	0.51 ± 1.24
S(T(J_Train)) (2673)	2.37 ± .19	3	0.046 ± 0.05	1.06 ± .23	3	0.25 ± 0.88
T(M_Train) (98)	4.85 ± .50	7	0.716 ± 2.74	1.89 ± .72	7	2.65 ± 2.96
S(T(M_Train)) (1596)	4.29 ± .37	6	0.164 ± 0.10	1.30 ± .48	5	0.88 ± 1.85
T(J+M_Train) (159)	3.42 ± .58	5	0.370 ± 1.55	1.47 ± .68	6	1.29 ± 2.29
S(T(J+M_Train)) (2440)	2.23 ± .07	2	0.043 ± 0.05	1.04 ± .19	2	0.25 ± 0.85
2G(S(T(J+M_Train))) (5253)	**1.82 ± .07**	1	**0.036 ± 0.06**	**1.01 ± .10**	1	**0.21 ± 0.83**

Results are indicated in Table 4.4. As in the previous table, '*S(T_J_train)* (2673)' refers to the student trained on the 2673 audio-annotation pairs obtained with *T(J_Train)*. We estimate the average $offset_d$, fr_d and the mean rank of each SVD. The mean rank averages over songs the indiviual rank of each system for each song i.e. how far they are from the ground truth value. For instance, four systems a, b, c and d with offset deviations 0.057, 0.049, 0.057 and 0.063 seconds are ranked as: b = 1st, a = 2nd, c = 2nd and d = 3rd, respectively. The mean rank per model is the average of all individual ranks per song. Finally, the *position* value is the result of classifying the systems according to their mean rank value.

For this experiment we observe the same tendency than before: students outperform their teacher and the *Second Generation* is the one that achieves the best results.

Baseline SVD. The main motivation to improve $\hat{p}$ is that the alignment we observed with the baseline SVD systems was not good enough. This experiment quantifies this judgment. The T(*M_Train*) and T(*J+M_Train*) are ranked last in finding both the right offset and frame rate. Their values are considerably different from the ground-truth which produces an unnacceptable

alignment. Remarkably, the T(*J_Train*) is much better and its results are comparable to the student networks.

First students. Each student exceeds its teacher with consistently higher rank and lower deviations. S(T(*J+M_Train*)) is the best student. This is surprising because we use not particularly well-aligned data to train it (its teacher T(*J+M_Train*) is placed 5th and 6th for offset and fr). Yet it scores almost as well as the S(T(*J_Train*)), which was trained with better-aligned data (its teacher is the best one). We presume an error tolerance in the singing voice detection task. This tolerance is not critical when below an unknown value, but crucial above it: S(T(*_M_Train*)) (trained with the most misaligned data) is far worse than the other students.

Second student. As before, the *Second Generation 2G(S(T(J+M_Train)))* is the best system. It is placed first for both rankings and has the lowest deviation. However, the increase is moderate. We presume that we are reaching the limit of the alignment precision that we can achieve with the *NCC*, Section 4.4.

These results, together with the ones at Table 4.3, prove that DALI is improving at each iteration.

4.7 The multitracks

The multitrack version of DALI was built differently. In the WASABI project, there are 2k multi-tracks for popular music. The intersection between this set and the retrieved annotations is 863 multi-tracks. Nevertheless, these multi-tracks do not have any kind of standardization meaning that there are cases in which the sources are grouped by RAW files, other by STEMS files[14] or even several STEMS together in a track. Additionally, they come without any metadata or label, just the name in the form track_1, track_2, ..., track_n making difficult to work with them.

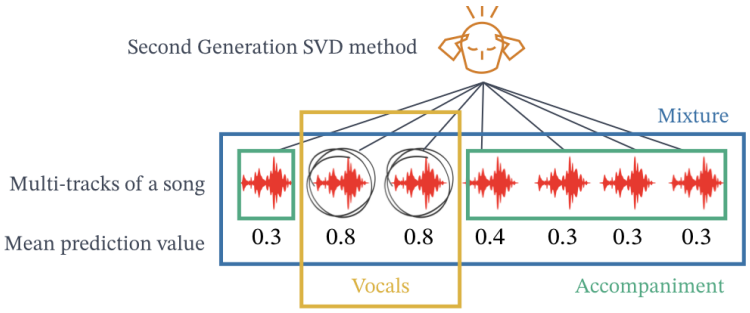


Figure 4.10: Method used for creating the *vocals*, *accompaniment* and *mixture* version

Since in Chapter 9 we focus on the vocal source sepration, we need to have three sources: *vocals*, *accompaniment* and the *mixture*. To do so, we use our best SVD, the Second Generation (see

[14]While RAW files define individual tracks in the mixture, for instance, each part of a drum kit is stored in a different file, STEMS files merge all the RAW files of a given instrument into a single file.

Section 4.5.4) as follow. Given all the tracks of a song, we apply this model to each one computing a different $\hat{p}$. Assuming that there must be at least one track with vocals (this subset of the multi-tracks is overlapped with DALI) we compute the mean of each $\hat{p}$. Finding the maximum of those mean and using a tolerance around this value (2%) we can find the tracks that include vocals. We then sum those tracks to produce the *vocal* one. The rest of the tracks are joined to define the *accompaniment* and all the tracks are merged to produce the final mixture (see Figure 4.10). Lastly, we align the annotations using the procedure describe in Section 4.4.

We manually verify the resulting sources. We have found that only 512 multi-tracks were correct. Out of the 351 wrong ones, there were 30 where the original tracks were cut in the middle of the song and 321 where the original tracks did not contain the vocals isolated but rather mixed with other instruments (usually guitar or keyboards). In the latter group, 69 songs are mostly good but vocals are combined with something else at the end of the track (when the chorus is repeated adding new instruments), 128 are mixed with an instrument in the background during the whole song and 154 are mixed with other instruments with a loudness similar to the vocals.

For future versions, we plan to prepare the rest of the instruments so that we can work with all the instruments that form the *accompaniment* section. This will allow us to deepen more into the work done in Chapter 8.

4.8 Conclusion

In this chapter, we explained our methodology to create the DALI dataset. We defined a loop where dataset creation and model learning interact in a way that benefits each other. Our approach is motivated by the Teacher-Student paradigm. The time-aligned lyrics and notes come from Karaoke resources where non-expert users manually annotated the lyrics as a sequence of time-aligned notes with their associated textual information. From the textual information, we derived the different levels of granularity. We then linked each annotation to the correct audio and globally align the annotations to it using the NCC on the VAS and SVP. To improve the alignment, we iteratively updated the SVD using the teacher-student paradigm. Through experiments, we showed that the students outperform the teachers notably in two tasks: alignment and singing voice detection. In our case, we showed that, in the context of deep learning, it is better to have imperfect but large datasets rather than small and perfect ones.

DALI is a great challenge. It has a large number of imperfect annotations that have the potential to make our field move forward. But we need to solve the issues still presented in the dataset. This requires ways to automatically identify and quantify them (see Chapter 6), which is costly and time-consuming. On the other hand, we have deep learning models with imperfections that are difficult to quantify. Therefore in DALI, we deal with two sources of imperfect information. This puzzle is also common to other machine learning domains. Our solution creates a loop where machine learning

models are employed to filter and enhance imperfect data, used then to improve the models. We prove that this loop benefits both: the model creation and the data curation. We believe that DALI can be an inspiration to our community to not regard model learning and dataset creation as independent tasks but rather as complementary processes.

However, there is room for improvement. In this first iteration of the DALI dataset, our goal was to find the correct audio candidates and globally align the annotations to it. But, we still have false positives. Moreover, once we have a good global alignment, we can try to solve local issues due to errors in the annotation process (see Chapter 5).

Chapter 5

Errors in DALI

So far, we can only guarantee that each new DALI version has better audio-annotations pairs and a more accurate global alignment than the previous one. We can also indirectly confirm that the current annotation timing (t_k^0, t_k^1) is good enough to create a state-of-the-art SVD system. But the dataset is still quite noisy and we know very little about the type of errors that are there. After a manual analysis in detail, we classify those errors in two groups: **global** errors that affect the song as a whole and **local** errors to non-professional users. In this chapter, we analyze these errors, propose solutions, suggesting some ways of indirectly measure the quality of the annotations based on a set of proxies.

5.1 Global errors

These errors affect the song as a whole and are usually due to problems in the transformation of the raw data (see Section 4.1) and the global alignment technique (see Section 4.4). They are the least frequent issues. Additionally, during our inquiry we have found almost no false positives meaning that the final audio we kept is the right one. On the other hand, there are still many true negatives that are not chosen due to the restrictive threshold we are using $T_{corr} = 0.8$. There is the possibility of using a less restrictive threshold. However, we do not know how this affects the number of false positives.

5.1.1 Global errors in time

The most common global time errors are those which have misaligned sections despite the audio-annotation pair having a high NCC and good $\hat{o}$ and $\hat{fr}$ values. In these cases, each section has a different offset. This can be solved by simply using the same NCC technique but instead of applying it to the whole audio track, doing it so locally for each paragraph. Thus, we isolate each $vas(t)$ of

$A_{paragraphs}$ and the corresponding $\hat{p}(t)$ segment around the corresponding time extract. We perform the NCC described in Section 4.4 on the two new vectors.

Moreover, there are songs with one or more missed sections. This is likely to occur when several vocalists sing at the same time or when a chorus is repeated at the end of the song but this is not indicated in the lyrics. The solution to these errors are not tackled and remain for future work. We just mark the lines that have possible errors as described in Section 5.1.3.

5.1.2 Global errors in frequency

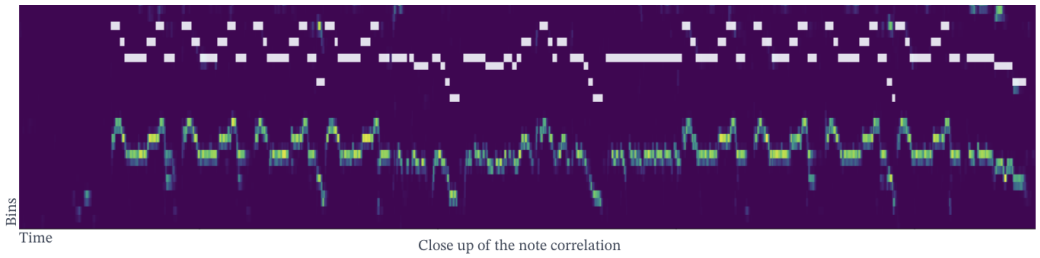


Figure 5.1: Annotations as matrix overlapped with the f_0. Note how the annotations are shifted in frequency by a constant factor.

The raw annotations store the notes as interval differences with respect to an unknown reference frequency (see Figure 4.1). Most of the songs use C4, which is the reference used for transforming these differences into frequencies (see Section 4.1). But, this is not always the case and some songs use a different reference as shown in Figure 5.1.

To solve the global frequency errors, we perform a new correlation in frequency between the note level $(a_{k,notes})_{k=1}^{K_{notes}} = (t_k^0, t_k^1, f_k, l_k, i_k)_{notes} \rightarrow (t_k^0, t_k^1, f_k)_{notes}$ and a f_0 extraction (Doras et al., 2019), that extracts the main pitch for each instant. Pitch defines how we perceive the periodicity of music signals and it is highly related to the notion of "musical notes". It permits us to classify musical events as high or low. The extracted f_0 is a matrix over time where each frame stores the pitch likelihoods obtained directly from the original audio. We compress the original f_0 representation to 6 octaves, 1 bin per semitone and a time resolution of 0.058s. Similar to the process done in Section 4.4, we then transform the annotations $(a_{k,n})$ into a matrix. Unlike the previous correlation, we are measuring correlation along the frequency axis and not in the time one. We then simply transpose all the f_k in the $a_{k,note}$ by the same value. The transposition factor covers all the frequency range in the f_0 matrix. We find the frequency factor that maximizes the energy between the annotated frequencies $a_{k,note}$ and the estimated f_0. This defines the correct global position of the annotations of the whole set $S = \{A_{notes}, A_{words}, A_{lines}, A_{paragraphs}\}$.

Besides, we calculate the new "flexible" versions of the melody metrics *Overall Accuracy, Raw Pitch Accuracy, Raw Chroma Accuracy, Voicing Recall, Voicing False Alarm* (Bittner and Bosch,

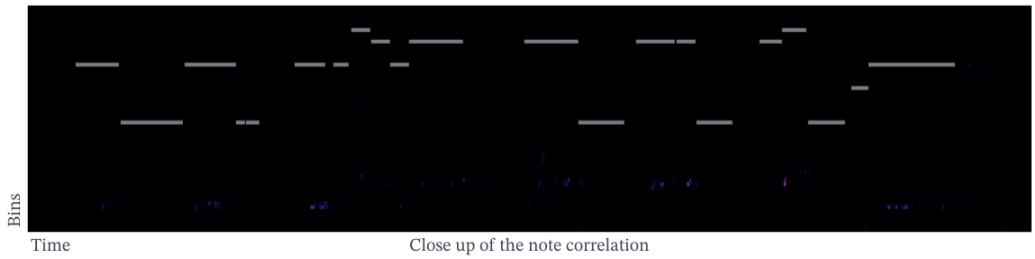


Figure 5.2: Overlap between DALI annotations (white lines) and the f_0 matrix with pitch likelihood distributions. Although the annotations are correct, the system that computes the f_0 does not find any f_0 in this region (the black background means low likelihood for all the possible f_0). This results in a low-frequency correlation.

2019). These metrics add new extra knowledge for understanding the quality of the annotations. Together with the NCC, these can guide us to know which annotations are good and which ones are of lesser quality. We can assume that a high *Raw Pitch Accuracy* suggests a good alignment. However, low metrics do not necessarily indicate a bad one since this can be due to errors in the f_0 extraction (see Figure 5.2).

5.1.3 Multitracks

The most usual mistakes are some vocal segments without any annotations or some annotation lines in silent segments. To find vocals segments without annotations, we simply look for audio segments without annotations that have high energy in the isolated **vocals**. We perform the inverse process to find the annotations lines in silence segments. That is, to look for audio segments with annotation $a_{k,lines}$ that have low energy. We add those areas to the original annotations to be taken into account in future chapters. There are on average 0.26 $a_{k,lines}$ on possible silence segments and 2.169 audio segments with vocals (usually ohs) without annotations.

5.2 Local errors

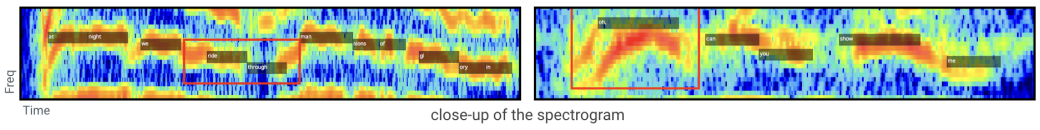


Figure 5.3: Type of errors still presented in DALI. [Left] Mis-alignments in time. [Right] Mis-alignments in frequency. These problems are difficult to detect.

Local errors occur because the karaoke users that did the anotations are non-professionals. They cover local segment alignments, text misspellings and note frequency errors (see Figure 5.3):

- **Time**: errors in the positions of the start (t_k^0) or end (t_k^1). Notes are placed in the wrong position in time or have the wrong duration.
- **Frequency**: errors in f_k. These errors are quite arbitrary, including octave, semitones and other harmonic intervals such as major third, perfect fourth or perfect fifth.
- **Text**: misspellings in l_k. Additionally, there are also errors in the phoneme level due to the automatic process employed (Chapter 4.1).
- **Missing notes or melodies**: notes can be annotated where there are no actual notes in the audio, and conversely, notes in the audio can be missed all together (during humming, ohs or similar parts).

Local errors are difficult to detect, and they can cause every note event to be wrong in some way. At the individual time-frame level, notes with incorrect start/end times will have errors at the beginning/ending frames, but can still be correct in the central frames. To quantify such local issues, it would be necessary to manually review all the annotations one by one. This is demanding and time-consuming. Indeed, this is what we aim to avoid. Beside, correcting them requires expert knowledge and doing it manually is unfeasible (for one song there are on average 372 notes).

We address local errors concentrating on the note level $a_{k,notes}$. The solutions found for this level will be then propagated to the rest of levels. For the following experiments, we discard their text dimension l_k, $(a_{k,notes})_{k=1}^{K_{notes}} = (t_k^0, t_k^1, f_k, l_k, i_k)_{notes} \rightarrow (t_k^0, t_k^1, f_k)_{notes}$. Nevertheless, this dimension can give many useful insights for future work. Given the current set of notes a and the audio signal x, we aim to find a function $g()$:

$$g(a, x) = a' \approx a* \tag{5.1}$$

where a' is a new annotation expected to be close to the hidden correct one $a*$. The a' has to have the same number of elements as a and $g()$ can only modify the values (t_k^0, t_k^1) of each note without overlap between notes.

5.2.1 Local alignment

Our task is to compute local modifications on the time series $(a_{k,notes})_{k=1}^{K_{notes}}$ to find a monotonic[1] alignment where the distance between a and x is the smallest. We target it using alignment techniques. These aim to order sequences so that we can identify regions of similarity.

After a first exploration of the different techniques, we decide to deepen in the Connectionist Temporal Classification (CTC) loss (Graves et al., 2006). This loss computes internally a conditional probability $p(y|x) = \sum_{a \in A_{x,y}} \prod_{t=1}^{T} -log(p_t(a_t|x))$[2], where y is the **sequence of states** to align,

[1] Advances in time in the signal correspond to the same position or advance in the target sequence

[2] The loss then propagates the error to update x to maximize the alignment.

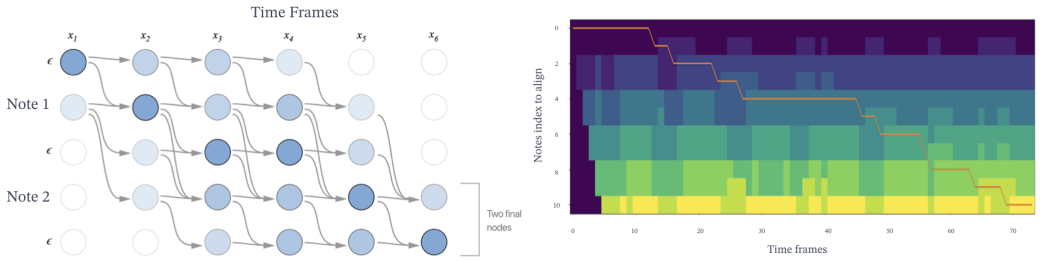


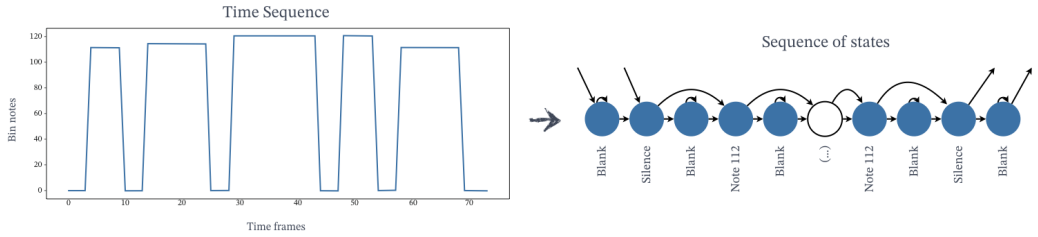
Figure 5.4: [Left] Graph alignment example with all the valid alignment paths. In the y-axis, the target notes to align (a sequence of two notes). In the x-axis the time evolution of the observation probability x with six-time steps. The intensity of each node represents the value in x for the target state at a time t. The blank state ϵ can go either to itself or the next state. On the other hand, no-blank states can go either to themselves, the next blank state or the next no-blank state. Reprinted from `https://distill.pub/2017/ctc/`. [Right] A real accumulation matrix obtained with Viterbi and the final path that minimizes the cost obtained after backtracking.

x the **observation probability** matrix of each state at a given time t and $\sum_{a \in A_{x,y}}$ all the valid alignments of y given x. We find these alignments with the Viterbi algorithm (Ryan and Nudd, 1993), a dynamic programming technique that evaluates the best way to arrive at each node at a given time. This results in an accumulation matrix that can be backtracked to find the best alignment. The **observation probability** x describes the audio as state probabilities over time and is obtained in such a way that we can direct evaluation $p(y_t|x_t)$ at a particular node. The **sequence of states** is an ordered set of states (e.g. a phrase like 'This is an example'), drawn from a finite and limited alphabet (e.g. the letters usually with a white-space character). CTC adds the blank state to the alphabets for codifying repetitions, allowing to distinguish between consecutive states with the same value. This formalization has interesting advantages. It describes the signal to be warped (y) as a graph with defined states and possible transitions. It requires a probability formalization to directly evaluate $p_t(a_t|x)$ where one signal is described a conditional probability of the other. Finally, it scales well to long sequences.

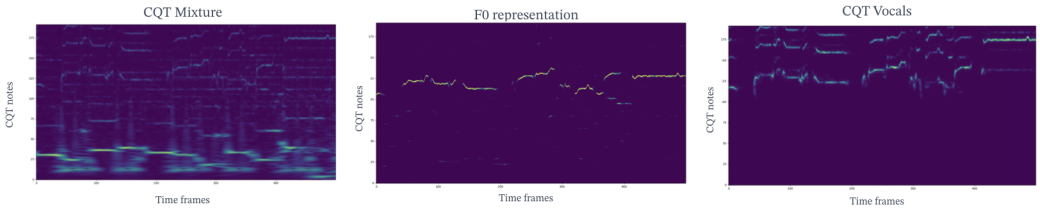
Figure 5.4 illustrates the process. At x_1, we evaluate all the possible initial states in y (either the first blank character or the note 1). We advance in time to x_2. Starting at the positions defined by the previous points, we evaluate the possible advances in y (if we are in a blank state we can either repeat the state or move to the next state, if we are in note 1 state we can repeat the state, move to the next blank state or the next note). We reproduce this process iteratively until covering x. At a given time-step, a state can be linked to states with several previous steps. To optimize this process, we compute an accumulation matrix using Viterbi where we keep only the link that defines the most probable path to arrive at one state at a particular time. In the end, we obtain the most promising alignment of y in x backtracking this matrix.

Thereby for our problem, we need first to transform $a_{k,notes}$ into a graph, where each state is the frequency information f_k (quantized to bins notes) and the silence state (see Figure 5.5a). We

add the blank state between states and limited the transition between note states so that a note can only go to a blank state or the next note but not repeat itself. This does not affect the results and simplifies the computation. Note how the graph removes the original time information (t_k^0, t_k^1) contained in the annotations. This reduces the length of the sequence which is faster and easier to control. We need then to define the **observation probability**. A natural choice is to obtain the f_0, where the audio is described as likelihoods over time of the fundamental frequencies. To do so, we first artificially create the silence likelihood at each time step by computing $1 - \max_{bins} f_0(t)$. We then apply a $softmax$ so that the sum of all the bins (plus silence) at each time step is 1. Using this formalization we can easily compute new alignments using $g(a, x) = a'$.



(a) We transform the note annotations into a sequence of states, losing the duration. In between these states, we add the blank state. This state encodes the repetition of the previous step, allowing distinctions between two consecutive notes with the same value. In this particular example, there is always a silence between notes but it is not always the case.



(b) There are many possible observation probability matrices for performing the alignment.

Figure 5.5: The main elements for alignment used are: [Top] the observation probability matrix of each state at a given time. [Bottom] A sequence of states to align.

Figure 5.6 illustrated two new a'. In the first example, we can observe how the new annotations seem to be more precise than the original ones. However, analyzing the path (orange small line) we realize that the first note is compressed radically and the second one covers the original duration of both notes. This effect is even more severe for the 3rd, 4th and 5th states. Here, the first note covers all the duration and the other two are compressed to a minimum value. In contrast, the same experiment was carried out with Dynamic Time Warping (DTW). Here the effect is more dangerous because it merges notes with the same f_k making it impossible to distinguish consecutive ones. The second example proves how errors in the observation probability, x can induce dangerous missalignments. The f_0 estimation misses some essential notes which distorts the whole alignment.

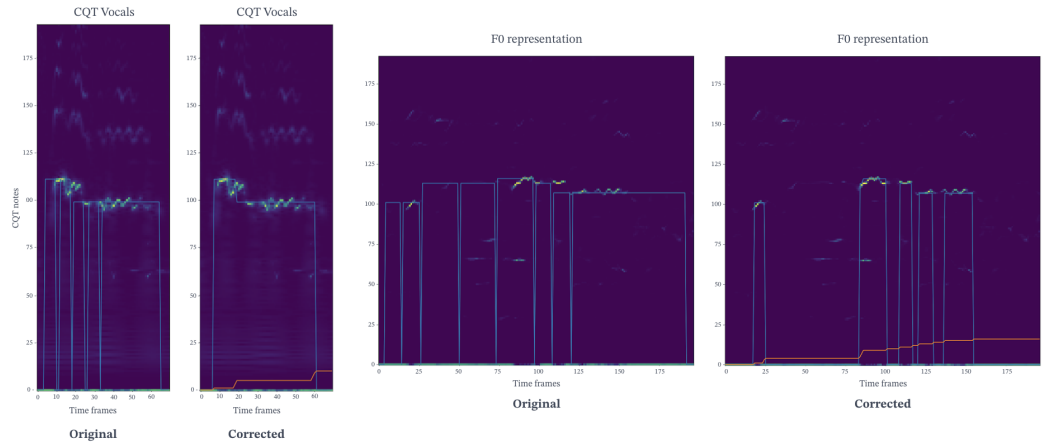


Figure 5.6: [Left] Example of how errors in the observation probability, x can induce dangerous missalignments. [Right] First alignment example where the new a' seems to be better than the original. Nonetheless, some new notes are shorter that they should be.

Alignment configurations.

As shown in Figure 5.6, we have tested many configurations. We can use as **observation probability** many different audio representations x (see Figure 5.5b). Appart from the f_0, we can directly use the normalized log magnitude Constant-Q transform (CQT) (Brown, 1991) as a likelihood function. CQT provides suitable representation of a musical signal without introducing possible errors. The frequency bins of the CQT are logarithmically spaced and with equal center frequencies-to-bandwidth ratios (Brown, 1991) (see Figure 5.7). Formally, the center of each bin $f_b = f_{min}2^{b/p}$, where f_{min} is the lowest frequency we aim to capture, b the number of bins per octave. For music signals, it is common to set p to a multiple of 12 because there are 12 semitones per octave in the Western music scales. Hence, each f_b corresponds to a semitone frequency. The bandwidth of each filter p^{th} is defined as $\triangle f_p = f_p(2^{1/b} - 1)$, producing constant center bin-to-bandwidth ratio $Q = \triangle f_p / f_p$. The CQT produces complex-valued coefficients. We work the mixture audio or the isolated vocals[3] (Jansson et al., 2017). All the values of normalized CQT are between 0 and 1. This can be seen as a pseudo-likelihood function, that we transform into probabilities as before (artificial silence + softmax). The only difference is that the silence is computed on the isolated vocals for both CQTs (mixture and vocals).

We can also combine the f_0 and the CQT into a single observation probability matrix. Since both have the same dimensions, we do an element-wise mean. We usually use more than 12 bins per octave to have a fine frequency resolution. We need to define tolerance in the annotation to evaluate observations x_t to define the final contribution. Since real notes do not stay constant in frequency but rather evolve with variations around the central frequency and annotations define single bins, to

[3]Source separation techniques are treated in detail in Chapter 9.

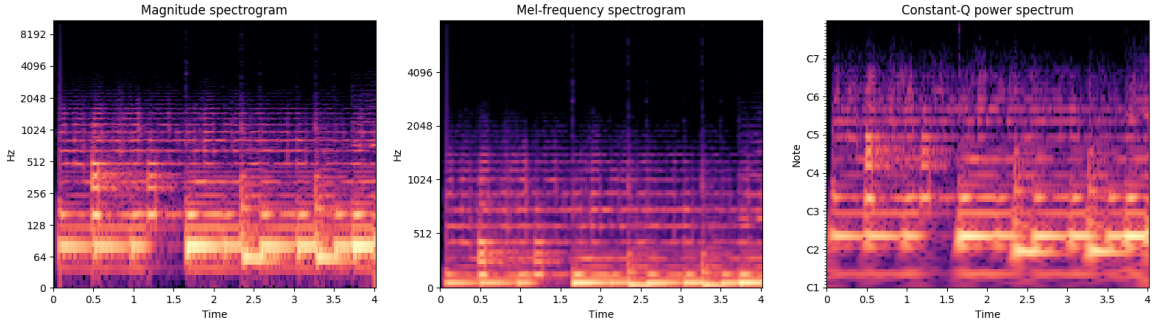


Figure 5.7: The three frequency domain transformations used in this thesis for an illustrated example extracted from a 4-seconds audio segment.

properly evaluate $p(y_t|x_t)$. A small tolerance results in short notes with long silences. On the other hand, a large tolerance artificially creates long notes. Additionally, to be robust to common octave errors in individual notes, we can add the contribution of each octave bin below and above. This can be extended to reduce the states and **observation probability** into a single octave, focusing only in the correctly class note without its pitch.

Finally, we have formalized the problem focusing only on the note information. But, we can also use the sequence the text information l_k in $(a_{k,notes})_{k=1}^{K_{notes}} = (t_k^0, t_k^1, f_k, l_k, i_k)_{notes} \rightarrow (t_k^0, t_k^1, l_k)_{notes}$ as target states. The observation probability matrix will be then the character information obtained directly from the audio (Stoller et al., 2019). Another different configuration is to combine both dimensions, where each state is defined by (l_k, f_k). We will need also to consider matrices with two different representation of the audio signal x, one for audio and one for text information.

Instead of obtaining a single alignment using all the notes in $a_{k,n}$, we can do individual alginments of smaller segments, e.i. computing $p(y|x) = \sum_{A \in A_{x,y}} \prod_{t=1}^{T} p_t(a_t|x)$ using as target y the sequences define by each element in the different levels of hierarchy. Thereby, we compute as many alignmets as lines $a_{k,l}$ has, using for each one the notes contained in a line. We have done this also using also the paragraph level $a_{k,paragraph}$. The simplicity of the alignment allows many further modifications in its computation:

- We can weight the observation matrix x by the annotation a i.e. introducing penalties to observations that a given time t differs from the original annotations, a_k. This is done by transforming $a_{k,n}$ into a matrix (with one at the note positions and the desired weight elsewhere) that is multiplied with the observation matrix x.

- We can also try to solve the errors in the states y. This is done using a beam search decoding[4] that creates different versions of the sequence of states y (with variations such as major third, perfect fourth or perfect fifth) at each time t, keeping only a fixe number of possible paths for

[4] A tree algorithm that searches many (a fixed number of beams) potentially optimal paths in parallel

future examination.

- Since this alignment is a Markovian process, we can introduce weights in the transition between states according to the duration of each note. That means that every time we are evaluating the best way to arrive at a node state for creating the accumulation matrix, we can multiply the contribution of each linked states by a factor. These factors are defined by the relative duration of each note in the sequence y. This integrates the duration of the notes.

- Another way to integrate note durations is by adding another penalty to favor paths that produce notes with similar durations as the ones in the a_k. When computing the backtracking we evaluate several alignments (not only the most promising one) favoring the ones with similar notes durations. This helps in avoiding conflict transitions that result in notes with a very distinct duration from the annotated one.

Although these variations are quite powerful and produce interesting results. Nevertheless, it is quite challenging to establish which one works the best, i.e. which a' is really the closest to the hidden correct $a*$. We have analyzed manually many configurations and we cannot conclude with certitude which one is the best. For some examples, a particular configuration seems to help but it introduces errors for others. Quantifying how the accurateness of the new a' is challenging and still requires many manually reviewing (annotations have to be analyzed one by one every time we have a new configuration). This is extremely demanding and time-consuming and not scalable. We need then to address a new research question: **Is the new a' better than before? How much? Which $a_{k,n}$ are getting better and which worse?**. This is the central topic of the Chapter 6.

5.3 Conclusion

In this chapter, we analyzed and improved over the errors presented in DALI. The types of issues are quite diverse and difficult to quantify. We introduced solutions to globals time errors that still persist as well as a correlation based on the note annotations and the f_0 for solving the global frequency shifts. This correlation can be used as an extra insight into the quality of the annotations. Local errors are a great challenge. They do not have constant types of corruption nor class dependencies. We explored the internal CTC alignment technique as a solution to solve them. The performance of this technique is heavily affected by factors such as the **observation probability** used and its adjustable configurations. However, we cannot measure if the new annotations are better or worse. In the next Chapter, we explore an automatic way to measure the quality of the annotations that can be used to evaluate each new a'.

Chapter 6

Training with noisy data[1]

Labeled data is necessary for training and evaluating supervised learning models, but the process of creating labeled data is error prone. Labels may be created by human experts, by multiple human non-experts (e.g. via crowd sourcing), semi-automatically, or fully automatically. Even in the best case scenario where data is labeled manually by human experts, labels will inevitably have inconsistencies and errors. The presence of label noise is problematic both for training and for evaluation. During training, it causes models to converge slower (requiring much more data) or overfit the noise, resulting in poor generalization. Label noise in evaluation sets results in unreliable metrics with artificially low scores for models with good generalization and artificially high scores for models which overfit data having the same kind of noise. This is also the case of the DALI dataset. Due to its nature (non-expert user annotations) and the way it was created (see Chapter 4), the assigned labels are considerably noisy. Having control over the errors is essential to be able to work with databases like DALI. This allows us to know which data points are likely wrong or correct. We can then correct them or select only the correct points. Nevertheless, manual quantification is demanding and time-consuming.

In this chapter, we present a **self-supervised data cleansing** model which exploits the structured nature of music annotation data in order to predict incorrectly labeled time-frames. This directly addresses the question of ***how good are the annotations in DALI?*** and aim to establish a similarity relationship between labels and the audio signal. The proposed strategy is designed for multiclass label noise that is time and/or position-dependent. The model requires examples of correct and incorrect labels. We generate training data for this model by selecting a subset of data with labels we believe to be correct as the "correct" examples, and creating artifical deformations of these labels as the "incorrect" examples. This set is then used to train a self-supervised binary classifier that predicts the probability of having a wrong label. We show that the model can be

[1]The work reported here was done in collaboration with Rachel Bittner who helped in framing and formalizing the problem and trained the different transcription models used for the validation.

defined as a simple binary classifier (is the label correct or incorrect?), which can be learned with simpler architectures than those needed to solve the target task (e.g. a perfect note transcription model). We demonstrate the usefulness of this data cleansing approach by training a transcription model on the original and cleaned versions of the DALI, improving the performance by almost 10 percentage points. Finally, we outline several other potential applications of the proposed approach.

6.1 Working with noisy data in supervised learning

Supervised learning approaches commonly assume that once a dataset is created the assigned labels are unambiguous and accurate. Nevertheless, labels may be noisy, inconsistent, subjective or incomplete (Xiao et al., 2015; Reed et al., 2014; Laine and Aila, 2016). Many studies have shown that label noise has consequences for learning (Nettleton et al., 2010; Arpit et al., 2017; Zhang et al., 2016; Sukhbaatar et al., 2015; Frenay and Verleysen, 2014; Mishkin et al., 2017), including requiring increased complexity and deteriorating classification performance because models eventually also memorize these wrongly given labels. Thus, it is important to consider that the labels assigned to data points in a dataset may not be equivalent to the underlying true labels. Label noise is especially prevalent when using data sources from the Web (Mahajan et al., 2018), such as DALI.

The process that pollutes labels is referred to as **label noise**[2] (Frenay and Verleysen, 2014). Manual curation strategies for removing label noise from datasets are notably time-consuming (especially when domain expertise is required, like in our case). Moreover, they are not scalable. There are a number of automated methods that have been proposed for dealing with label noise, falling into two general categories (Frenay and Verleysen, 2014).

The first category is **Learning with Noisy Labels**, where methods are built to retain their classification performance in the presence of label noise. They explore strategies for filtering the noise or implicitly modeling it as part of the training process[3]. This approach is model-centric, where the overall goal is to maintain or improve the performance of the specific model. The second category is **data cleansing**, which is typically a pre-processing step where incorrect labels are identified and removed from the dataset or corrected.

The vast majority of data cleansing techniques involve training a model (or a subpart of a model) to predict which labels are most likely to be incorrect. This information is used then to remove or correct these data points. These methods suffer from the classic "chicken or the egg" problem; accurate filtering of noisy data requires an accurate model, but it is difficult to train an accurate model on noisy data. As a result, they typically suffer from either removing too many data points, throwing away good data (particularly points that are difficult -but important- to model) or letting

[2]Note that this is distinct from feature noise, which affects the value of the observation/features, i.e. the model input in contrast with label noise that concerns the model outputs (target to obtain).

[3]Note that this is distinct from approaches that, as a result of the label noise, learn inaccurate patterns damaging the performance. Approaches that implicitly model label noise want to have a defining part in the architecture that captures the noise contribution and remove it from the final output.

Table 6.1: Variables used for the data cleansing formalization.

Varaiable	Definition
$\mathcal{X}$	generic input space
$\mathcal{Y}$	generic output space
$\pi_{\mathcal{X},\mathcal{Y}}$	distribution that indicates if $\mathcal{Y}$ is an incorrect label for $\mathcal{X}$
$g(x, y)$	the probability that y is an incorrect label for x
$h : \mathcal{X} \rightarrow \mathcal{Y}$	generic classifier
S	a generic dataset with pairs (x_i, $\hat{y}_i$)
Z	a generic dataset with triples (x_i, $\hat{y}_i$, z_i)
I	set of all $(\mathcal{X}, \mathcal{Y})$ in a generic dataset
I^+	set of all ($\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$) in a generic dataset
F	a subset of I that contains all of the correctly labeled data points
i	index over instances in a dataset
x	a generic input instance
y	a generic true output instance
$\hat{y}$	a generic label instance
z	a generic label that indicates if $\hat{y} = y$

too many noisy datapoints through. Both cases result in reduced model performances (Nettleton et al., 2010; Arpit et al., 2017; Zhang et al., 2016; Sukhbaatar et al., 2015; Frenay and Verleysen, 2014; Mishkin et al., 2017).

Despite the clear detriment to downstream performance from training on noisy data, data cleansing techniques are not universally applied. We hypothesize that the lack of usage during training stems from (1) the (typically false) assumption that the labels for datasets are already correct and (2) the costly process of applying most data cleansing techniques since it is not just the output of one model but rather the output of several models or different configurations of the same model (Laine and Aila, 2016). Finally, the results of data cleansing are rarely fed back to the dataset itself, and the process needs to be repeated each time the data is reused.

Rather than training a model to fit the labels themselves, we propose a simple data cleansing approach that trains a model to predict if the label of a datapoint is correct or not. This can be performed with much simpler models (particularly for complex tasks); determining "is this the correct answer" is almost universally much easier than "what is the correct answer". This approach has the advantage that it is independent of downstream inference task. Additionally, it is easy to create data for which the labels are wrong, by simply distorting correct labels.

6.2 Label noise formalization

The problem of label noise can be formalized as follows (see Table 6.1 for a summary of the variables used in this section). Consider an input/output (data/label) space $\mathcal{X} \times \mathcal{Y}$ endowed with a probability measure $\pi_{\mathcal{X},\mathcal{Y}} : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]$, and a sample made of i.i.d.[4] points $(x_i, y_i)_{i=1}^m$ drawn

[4]Independent and identically distributed

from the distribution $\pi_{\mathcal{X},\mathcal{Y}}$ (Friedman et al., 2001). $\pi_{\mathcal{X},\mathcal{Y}}$ is equal to zero everywhere that $\mathcal{Y}$ is a correct label for $\mathcal{X}$, and one elsewhere i.e. it defines if a label y_i is a truth label for a data point x_i. In practice, we do not have access to the points (x_i, y_i) themselves (nor to $\pi_{\mathcal{X},\mathcal{Y}}$), but instead to a dataset $\mathcal{S}_I = \{(x_i, \hat{y}_i)\}_{i \in I}$, where $\hat{y}_i$ are possibly wrong labels and $I = \{0, 1, 2, \dots m\}$ is an index.

Thereby, any supervised learning task can be formalized as a classifier trained using $\mathcal{S}_I$:

$$h_I : \mathcal{X} \to \mathcal{Y} \tag{6.1}$$

6.2.1 Learning with Noisy Labels

This set of approaches for dealing with noisy labels considers the scenario where a classifier h_I is trained, using the full dataset $\mathcal{S}_I$ despite the noise. The methods are built to mitigate the effects of label noise, remaining accurate despite its presence.

Most of the approaches directly model the noisy distribution in the loss function. For example, by creating noise-robust loss with an additional softmax layer (removed during inference) to predict correct label during training (Goldberger and Ben-Reuven, 2017), or with a generalized cross-entropy that discards predictions that are not confident enough while training, looking at convergence time and test accuracy (Zhang and Sabuncu, 2018), or inferring the probability of each class being corrupted into another (Patrini et al., 2017). However, this restricts models to specific types of loss functions, noisy types or distributions, assuming that the noise definition is class dependent and conditionally independent of inputs given the true labels. Finally, models trained in this way usually need larger datasets to capture enough signal from the noise.

Semi-supervised learning uses reliable labeled data together with a large amount of unlabeled data (Zhu, 2005). Hereabouts, systems automatically leverage unlabeled/noise data through deriving insights from the labeled data. If you have a subset of data you can trust, even several noisy labels can be dealt effectively (Hendrycks et al., 2018). Recent ideas such as *consistency*, the ability of a model to "disagree" with target labels and re-labeling them during training (Reed et al., 2014) have been successfully implemented. *Consistency* can be also combined with pseudo-labeling on weakly-augmented unlabeled images, where the pseudo-labeled are only retained if the model produces a high-confidence prediction. The model is then trained to predict the pseudo-label when fed a strongly-augmented version of the same image (Sohn et al., 2020). Authors also use undirected graph to model the relationship between noisy and clean labels (Vahdat, 2017). The inference over latent clean labels is intractable and regularized during training.

6.2.2 Data Cleansing

The set of approaches aims to find a subset $F \subseteq I$, such that F contains all of the correctly labeled data points.

Let

$$g(x, \hat{y}) = \pi_{Y|X}(\hat{y}_i|x_i) \tag{6.2}$$

be the probability that $\hat{y}$ is the wrong label y for x. Then, the ideal goal of data cleansing is to find

$$F = \{i : i \in I, g(x_i, \hat{y}_i) = 0\} \tag{6.3}$$

That is to say, the subset F where all $\hat{y}_i = y_i$ (a true label for x_i). Given such a subset F, we can train classifier h_F without needing to account for label noise.

This approach has several advantages over learning directly with noisy labels. First, the filtering does not depend on the downstream inference task, thus one data cleansing method can be applied to filter data used to train many different models. Second, this allows downstream models to be trained with less complex models, as they do not need to account for label noise.

Outlier Detection

Outlier detection-based data cleansing methods use techniques for detecting outliers in data distributions to remove noisy labels. In general, they apply some form of anomaly measure $a_I(x, \hat{y})$ and filter out points where the anomaly measure falls above a threshold τ:

$$F \approx \{i : i \in I, a_I(x_i, \hat{y}_i) < \tau\}$$

For instance, measuring data complexity using its distribution density in feature space to alleviates the negative impact of the noisy labels (Sheng Guo and Huang, 2018).

Model Prediction

The vast majority of data cleansing methods are model prediction-based. In its simplest form, this category consists of training a model (or ensemble of models) $h_I(x_i) : \mathcal{X} \to \mathcal{Y}$ on the original dataset $\mathcal{S}_I$, and removing points where the label predicted by the model does not agree with the dataset's label:

$$F \approx \{i : i \in I, h_I(x_i) = \hat{y}_i\}$$

Note that in many cases, the choice of model h for data cleansing is often the same as the choice of model used after data cleansing. One of the new concepts successfully implemented is *self-ensembling*, which is the consensus on simulated predictions using the outputs of the network-in-training on different conditions (Laine and Aila, 2016).

Weakly supervised learning (Mintz et al., 2009; Mnih and Hinton, 2012; Xiao et al., 2015) deals with low-quality labels (or at a higher abstraction level than needed) to infer the desired target information. The teacher-student paradigm paradigm (Hinton et al., 2014) is the most employed

method[5]. In this case, the teacher is an extra network trained for selecting clean instances to guide the training of an extra network (a student) in a mentoring way (Jiang et al., 2018). It can be decoupled into two trained predictors and only update when they disagree (Malach and Shalev-Shwartz, 2017). This mitigates the problem of updating in wrong direction when label is wrong. Finally, co-teaching uses two nets that complement to each other (Han et al., 2018). At each mini-batch data, each network samples its small-loss instances as the useful knowledge, and teaches such useful instances to its peer network.

Active learning estimates the most valuable points for which to solicit labels (Settles, 2008; Krause et al., 2016). It has been applied for weighting important data points in loss during training to balance both noise and class imbalance problem. Weights are learned online during training to maximize performance on a small clean validation set (Ren et al., 2018).

All these ideas have assisted to achieve state of the art results when using noisy but large data. However, they rarely fed back the learnt knowledge to the dataset itself, and the process needs to be repeated each time the data is reused. Model are considerable more complex than the original downstream model and they usually learn a trade off between how much to trust the data and how much to refuse (Reed et al., 2014), throwing away good data or letting too many noisy datapoints through.

6.2.3 Position-Dependence

The most common and effective data cleansing approach is to build a model to identify and discard data points with incorrect labels. Most methods taking this approach do not assume any structure or correlation between different labels. Thus, the input/output space $\mathcal{X} \times \mathcal{Y}$ considers $\mathcal{X}$ to be multidimensional features and $\mathcal{Y}$ to be non-structured (e.g. as in multi-class classification). This is appropriate for many common tasks, such as image recognition, $\mathcal{X}$ are images (or features of images) and $\mathcal{Y}$ is a finite set of class labels.

However, in music, labels $\mathcal{Y}$ are often highly structured and time-varying, and the label noise is not random. For example, musical note-annotations, which we focus on in this work, are locally stable in time and follow certain common patterns. Typical noise for note events include incorrect pitch values, shifted start times, and incorrect durations, among others. In DALI, we face the case where $\mathcal{Y}$ is *position-dependent* - that is, where a label $y \in \mathcal{Y}$ is multidimensional, and varies as a function of the position relative to the corresponding $x \in \mathcal{X}$. This is also common in other tasks. For example, in object detection, $\mathcal{X}$ is the set of $(m \times n)$ images, and $\mathcal{Y}$ is an $(m \times n)$ matrix which is 1 inside the bounding box of an object and 0 otherwise. The position of the bounding box may not be well aligned with an object in the corresponding image. As another example, in speech recognition, $\mathcal{X}$ is an audio recording, which is a function of time, and its labels $\mathcal{Y}$ vary over time.

[5]Note how this formalization is different from the previous usage of this paradigm, a **semi-supervised** learning method where the teacher uses a different dataset than the original $\mathcal{S}_I$ to filter the data of student

The positions where these labels change in time define the boundaries of *segments*, which may be incorrectly aligned with the audio. Finally for the note annotations presented in DALI, *the time-frequency position* happens to be also the label information i.e. position in the y-axis corresponds to the note label.

6.2.4 Our Proposed Approach

We propose a model prediction based approach (Meseguer-Brocal et al., 2020a), but rather than training a classifier $d : \mathcal{X} \to \mathcal{Y}$, we directly train a model $g : (\mathcal{X} \times \mathcal{Y}) \to [0, 1]$ which approximates $h(x, \hat{y})$ from Equation 6.2. Specifically, we model F as:

$$F \approx \{i : i \in I, g(x_i, \hat{y}_i) = 0\} \tag{6.4}$$

Note that this is mathematically equivalent in the ideal case of the previous model prediction based approaches: given a perfect estimator h which always predicts a correct label y, $g(x, y) = \mathbb{1}_{h(x)\neq y}$ where $\mathbb{1}$ is the indicator function. However, for complex classification tasks with high numbers of classes and structured labels, modeling h can be much more complex than modeling g. For instance, consider the complexity of a system for automatic speech recognition, versus the complexity needed to estimate if a predicted word-speech pair is incorrect. Intuitively, you don't need to know the right answer to know if something is right or wrong. As an anecdotal example, consider the difference in difficulty of answering "Did they say 'apple'?" versus "What did they say?".

This idea is similar to the "look listen and learn" Arandjelovic and Zisserman (2017) concept of predicting the "corespondance" between video frames and short audio clips – two types of structured data. It is also similar to CleanNet Lee et al. (2018b), where a dedicated model predicts if the label of an image is right or wrong by comparing its features with a class embedding vector. However, this approach operates on global, rather than position-dependent labels.

We generate training data for g in a self-supervised way by directly taking pairs (x, y) from the original dataset as positive examples and creating artificial distortions of y to generate negative examples. In this section, we study the use of an estimator $g(x, \hat{y})$ for detecting local errors in noisy note-event annotations. See Figure 6.1 for an overview of the system.

6.3 Data Cleansing for the DALI dataset

In this section, we study the use of an estimator $g(x, \hat{y})$ for detecting local errors in noisy note-event annotations. See Table 6.2 for a summary of the variables used in this section, and Figure 6.1 for a top level view of the system.

Due to the nature of the data (non-expert user annotations) and the way DALI was created,

Table 6.2: Variables used for data cleansing note events.

Variable	Definition
L	the set of tracks in DALI
ℓ	index over tracks
I^ℓ	the set of data points for track ℓ
i	index over time frames
j	index over frequency bins
r_i	the time (s) at time frame i of the CQT
q_j	the frequency (Hz) at frequency bin j of the CQT
X^ℓ	the CQT of track l
$\hat{Y}^\ell$	binary note annotations matrix for track ℓ (Eq. 6.5)
X_i	time index i of X with all its frequency bins
$X_{a:b}$	a slice of X between time index a and b
$\hat{Y}_i$	time index i of $\hat{Y}$ with all its frequency bins
$\hat{Y}_{a:b}$	a slice of $\hat{Y}$ between time index a and b
D	the set of all time indexes in DALI (Eq. 6.8)
g	data cleansing model (Eq. 6.6)
$F \subseteq D$	subset of D based on g
$s(x_i)$	rough transcription model
$\kappa(\hat{Y}_i, s(X_i))$	agreement function (Eq. 6.10)
$\mu(\hat{y}_i)$	artificial modification function (Fig. 6.4)

Chapter 4; the note-event labels are considerably noisy. As detailed in Chapter 5, there are local errors in the positions of the start or end times of note events (notes are placed in the wrong position in time and/or have the wrong duration), as well as having errors in the labeled frequency value (see Figure 5.3). There are also extra labeled note events that do not correspond to any real event in the audio signal (e.g. during silence) and conversely, there are some real note events that are not labeled. The errors that we encounter in DALI typically do *not* include systematic substitutions of certain labels, and the positional noise is not label-dependent. Finally, these errors are time and position-dependent with respect to the audio signal. These type of noisy labels are common to other MIR datasets such as Raffel (2016). In DALI version 2, there are more than $2,8$ million note events with an average of 362 events per track. Correcting them requires expert knowledge and is very time consuming, making manual correction infeasible. Hence, this dataset serves as a perfect case study for our proposed data cleansing strategy.

As our input representation, instead of using the raw audio signal itself, we compute the CQT Brown (1991) as a matrix X, where X_{ij} is a time-frequency bin. The time index i corresponds to the time stamp $r_i = \upsilon \cdot i$ where υ is a constant defining the spacing between time stamps ($\upsilon = 0.0116$s for this particular task), and the frequency index j corresponds to a frequency q_j in Hz. The CQT is a bank of filters transformation centered at geometrically spaced frequencies. This results in a constant ratio of frequency bins ideal for musical purposes. We use a frequency bin resolution with 6 octaves, 1 bin per semitone, a sample rate of 22050 Hz and a hope size of 256, resulting in a time resolution of $11.6ms$. We compute the CQT from the original mixture and from

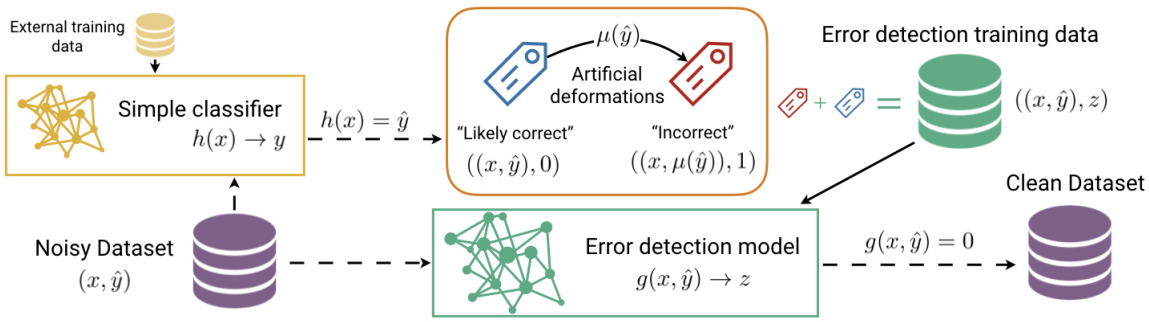


Figure 6.1: Top level overview of the proposed data cleansing approach.

the isolated vocal version derived from the mixture using source separation techniques Jansson et al. (2017) (see Chapter 8.3). We include the CQT of the isolated vocals to boost the information in the signal related to the singing voice, and couple it with the CQT of the mixture to include information we may have lost in the separation process.

We define the label target $\hat{Y}$ as a binary matrix created from the original note-event annotations. For a given track, let K be the set of note annotations, let t_k^0 and t_k^1 be the start and end time in seconds, and f_k be the frequency in Hz of note k. Then $\hat{Y}$ is defined as:

$$\hat{Y}_{ij} = \begin{cases} 1, & \text{if } t_k^0 \leq r_i \leq t_k^1,\ q_{j-1} < f_k \leq q_j,\ k \in K \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

Hence, $\hat{Y}$ has the same time and frequency resolution as X. We estimate the label noise at the ***time frame*** level, rather that at the pixel or note event level. Let $\ell \in L$ be an index over the set of tracks L, I^ℓ be the index of all time frames for track ℓ, and X^ℓ and $\hat{Y}^\ell$ be its CQT and label target matrices respectively. Let X_i^ℓ and Y_i^ℓ indicate a time frame of X^ℓ and $\hat{Y}^\ell$; we use all frequency bins j in the following discussion, so we index the data points according to their time index only. Finally, let $X_{a:b}^\ell$ and $\hat{Y}_{a:b}^\ell$ denote the sequence of time frames of X and $\hat{Y}$ between time indices a and b.

Our goal is to identify the subset of time frames $i \in I^\ell$ which have errors in their annotation for each track in a dataset by training a binary data cleansing model. Our data cleansing model is a simple estimator that can be seen as a binary supervised classification problem that produces an error probability function. Given a data centered at time index i, g predicts the probability that the label $\hat{Y}_i$ is wrong. Critically, we take advantage of the structured labels (i.e. the temporal context); as input to g we use $X_{a:b}$ and $Y_{a:b}$ to predict if the center frame $\hat{Y}_{(a+b)/2}$ of $\hat{Y}_{a:b}$ is incorrect. That is, we aim to learn g such that:

$$g(X_{a:b}, \hat{Y}_{a:b}) = \begin{cases} 0, & \text{if } \hat{Y}_{(a+b)/2} \text{ is correct} \\ 1, & \text{if } \hat{Y}_{(a+b)/2} \text{ is incorrect} \end{cases} \tag{6.6}$$

Thus, in order to evaluate if a label $\hat{Y}_i$ is correct using n frames of context, we can compute $g(X_{i-n:i+n}, \hat{Y}_{i-n:i+n})$. In the remainder of this work, we will define

$$g_n(X_i, Y_i) := g(X_{i-n:i+n}, \hat{Y}_{i-n:i+n}) \tag{6.7}$$

as a shorthand.

Let

$$D = \bigcup_{\ell \in L} I^\ell \tag{6.8}$$

be the set of all time indices of all tracks in L. Our aim is to use g to create a filtered index F, where:

$$F = \{i \in D : g_n(X_i, \hat{Y}_i) = 0\} \tag{6.9}$$

6.3.1 Training data

Let z_i be a binary label indicating whether the center frame $\hat{Y}_i$ for an input/output pair $(X_{i-n:i+n}, \hat{Y}_{i-n:i+n})$ is incorrect. To train g, we need to generate examples of correct and incorrect data-label pairs $\Big((X_{i-n:i+n}, \hat{Y}_{i-n:i+n}), z_i\Big)$. We will again introduce a shorthand $((X_i, Y_i), z_i)$ to refer to data points of the form $\Big((X_{i-n:i+n}, \hat{Y}_{i-n:i+n}), z_i\Big)$.

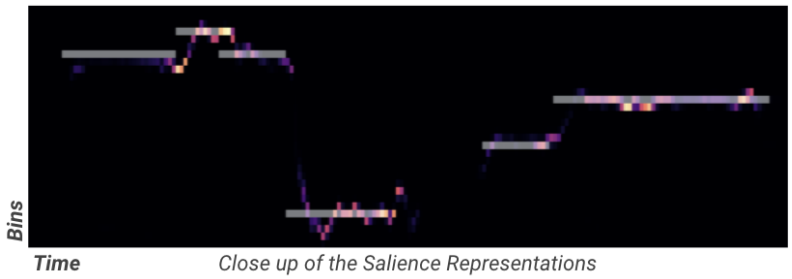


Figure 6.2: Overlap between DALI annotations (white lines) and the f_0 with pitch probability distributions.

Since we do not have direct access to the true label Y_i (indeed this is what we aim to discover), we first use a proxy for selecting likely correct data points. We first compute the output of a pre-trained f_0 estimation model $s(X_i)$ that given X_i outputs a matrix with the likelihood that each frequency bin contains a note Bittner et al. (2017). $s(X_i)$ is trained on a different dataset and has been proven to achieve state-of-the-art results for this task Bittner et al. (2017). $s(X_i)$ produces f_0 sequences, rather than note events, which vary much more in time than note events (see Figure 6.2), so we define an agreement function in order to determine when the labels agree. $s(X_i)$ is not a perfect classifier, and while its predictions are not always correct, we have observed that when the

agreement is high, $\hat{Y}_i$ is usually correct. However, we cannot use low agreement to find incorrect examples, because there are many cases with low agreement even though $\hat{Y}_i$ is correct. Therefore, we only use κ to select a subset of "likely correct" data points.

We compute both "local" (single-frame) and "patch-level" (multi-frame) agreement, and use thresholds on both to select time frames which are likely correct. The local agreement, κ_l is computed as:

$$\kappa_l(\hat{Y}_i, s(X_i)) = \max_j \left(\hat{Y}_{ij} \cdot s(X_i)_j \right) \tag{6.10}$$

and the patch-level agreement κ_p is a k point moving average over time of κ_l (see Figure 6.3). For the test set of g, we use very strict thresholds and select (X_i, Y_i) pair to be a likely correct if $\kappa_l > .999$ and $\kappa_p > .85$. For the training set, we use more relaxed thresholds, and select points with $0.9 < \kappa_l \leq 0.999$ and $0.7 < \kappa_p \leq 0.85$. This procedure gives us a set of positive examples in non-silent regions, but does not take into account the silent areas. In order to select correctly labeled points from silent regions, we take additional $(X_i, \hat{Y}_i)$ points from regions with low energy in the isolated vocals and no annotations in a window of length v. In this work we use $V = 200$ ($\approx 2,32$ s).

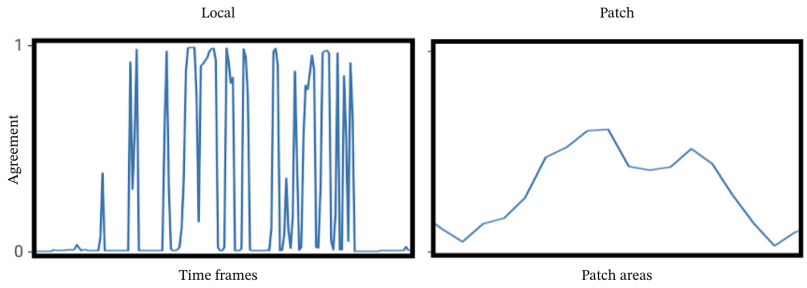


Figure 6.3: The two types of agreements $\kappa_l(\hat{Y}_i, s(X_i))$ used for selecting the $((X_i, \hat{Y}_i), z_i = 0)$ subset.

Once the "good" subset $((X_i, \hat{Y}_i), 0)$ is defined, we generate the "error" subset $((X_i, \hat{Y}_i), 1)$ where the Y_i is incorrect by artificially modifying $\hat{Y}_i$ in pairs from the "good" subset. These modifications $\mu(\hat{Y}_i)$ are not random but rather specific to match the characteristics of label noise presented in DALI (issues in the positions of the start or end times, incorrect frequencies, or the incorrect absence/presence of a note). These $\hat{y}_i$ should be contextually realistic, meaning that notes should have a realistic duration and should not overlap with the previous or next note (see Figure 6.4). The $((X_i, \hat{Y}_i), 1)$ data points for silent regions are generated by artificially adding random notes to data points where there is low energy in the isolated vocals. Finally, this gives us a dataset of $\{((X_i, Y_i), z_i)\}$ with which we can train g, created in a self-supervised way. The proposed process is summarized in Figure 6.1.

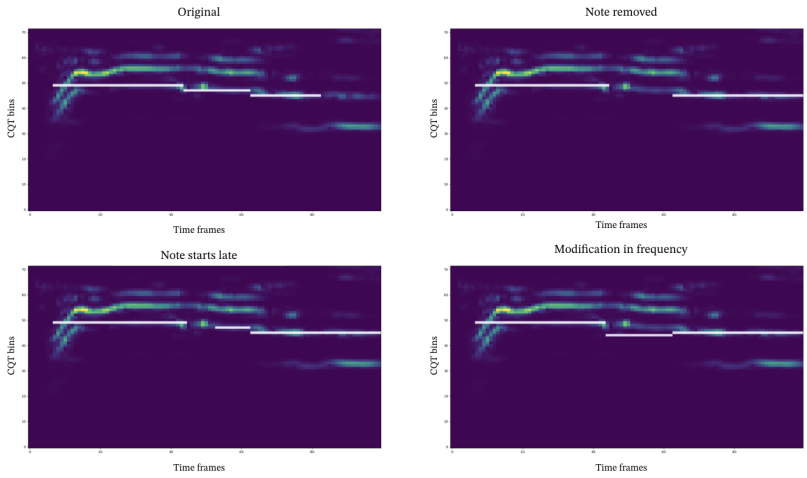


Figure 6.4: $\hat{Y}_i$ modifications for generating the fake wrong examples. These modifications are specific to match the characteristics of label noise.

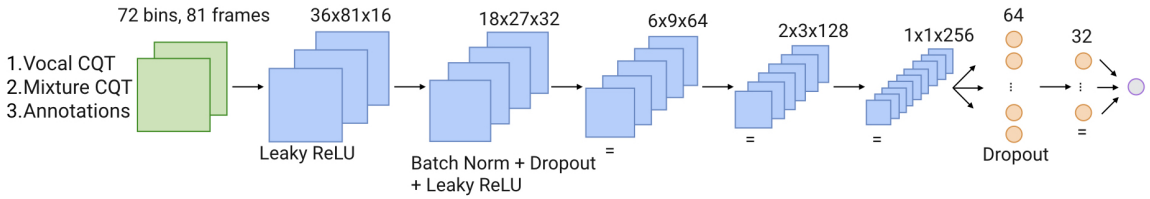


Figure 6.5: Error detection model for DALI.

6.3.2 Data cleansing model

We want to model the error probability function $g_n(X_i, \hat{Y}_i) = z_i$. In this work, we use $n = 40$. Hence, the input of the model is matrix with 72 frequency bins, 81 time frames (0.94 seconds)[6] and three channels: the two CQTs (mixture and vocals) $\{X_{i-n:i+n}\}$ and the label matrix $\{\hat{Y}_{i-n:i+n}\}$.

The proposed model is a standard CNN described at Figure 6.5. It has five convolutional blocks with 3×3 kernels, 'same' mode convolutions, and leaky ReLU activations for the first block and batch normalization, dropout and Leaky ReLU for the rest. The strides are $[(2,1), (2,3), (3,3), (3,3), (2,3)]$ and the number of filters [16, 32, 64, 128, 256] which generates features maps of dimension $36 \times 81 \times 16$, $18 \times 27 \times 32$, $6 \times 9 \times 64$, $2 \times 3x \times 28$, $1 \times 1 \times 256$. Then, we have two fully-connected layers with 64, 32 neurons, ReLu activation and dropout and a last fully-connected layer with only one neuron and sigmoid activation.

We test our approach using the second version of DALI Meseguer-Brocal et al. (2020b), which contains 7756 songs. The error detection model g is trained in a self-supervised way using the data generation method described in Section 6.3. The trained model g had a frame-level accuracy of

[6]For inference, the network would progress frame-by-frame.

72.1% on the holdout set, and 76.8% on the training set.

6.4 Validation

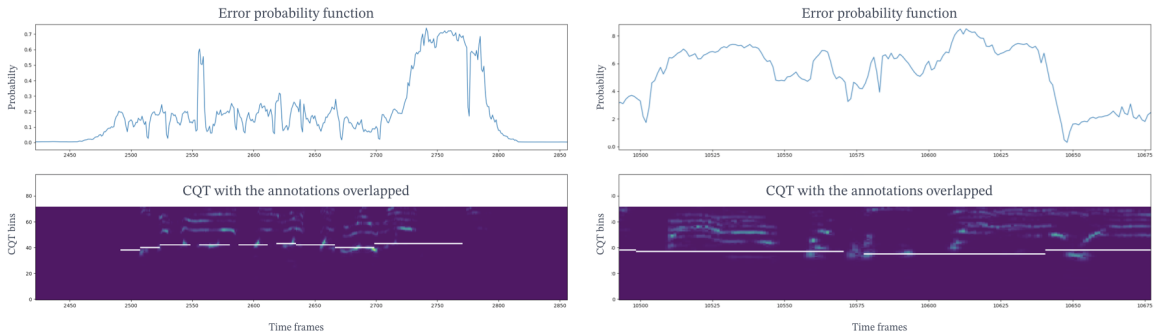


Figure 6.6: Error probability function examples. Each example contains the output of the error probability function and the annotations overlapped with the CQT of the vocals. [Left side] It also detects when the annotated note is not right. [Right side] The error probability function can spot issues in the duration of the last note and the beginning of the 4th note.

Validating the performance of g is challenging, as we only have likely correct and artificially created wrong examples, but we do not have any "real" ground truth good and bad examples. We first manually verified many random examples of the output of the error probability function, and found that appeared to be strongly correlated with errors in y_i. However, a manual perceptual evaluation of the error probability function is both infeasible and defeats the purpose of automating the process of correcting errors. Instead, we validate the usefulness of this approach by applying it to model training. In this section, we address the question: **Is error probability function useful? How much?**

The ultimate goal of a data cleansing technique is to identify incorrect labels and remove them from the dataset in order to better train a classifier. Thus, one way to demonstrate the effectiveness of the data cleansing method is to see if training a model using the filtered dataset results in better generalization than training on the full dataset.

We test this hypothesis in the context of the f_0 estimation. This task is a well-established problem in the MIR community with ground-truth datasets and proven architectures. It is also a task that can benefit from the type of errors the error probability function locates (errors in t_k^0, t_k^1 and f_k). The $a_{k,n}$ annotations can be easily transformed into pseudo-fundamental frequency annotations, defining a frequency bin over a set of frames that belong to the same note. Note how they differ from the traditional f_0 annotations, that follow the variations around the central frequency. Besides, the $a_{k,n}$ are only for vocals. For this reason, the final model aims to solve a slightly different and more complex task, vocal note transcription. We assume that this is constant for all our experiments.

Dataset	Scores	Training data used					
		All data		Filtered		Weighted	
MedleyDB	Raw Pitch Accuracy	0.403 ± 0.148	0.391	0.453 ± 0.143	0.464	0.495 ± 0.141	0.510
	Raw Chroma Accuracy	0.456 ± 0.138	0.443	0.502 ± 0.134	0.505	0.540 ± 0.131	0.532
iKala	Raw Pitch Accuracy	0.413 ± 0.101	0.416	0.484 ± 0.090	0.483	0.535 ± 0.092	0.545
	Raw Chroma Accuracy	0.441 ± 0.094	0.438	0.515 ± 0.088	0.520	0.546 ± 0.088	0.551
Both	Raw Pitch Accuracy	0.411 ± 0.112	0.414	0.478 ± 0.103	0.482	0.527 ± 0.105	0.542
	Raw Chroma Accuracy	0.444 ± 0.104	0.439	0.513 ± 0.099	0.515	0.545 ± 0.098	0.551

Table 6.3: Detailed results with the mean, standard deviation and median for the Raw Pitch Accuracy and Raw Chroma Accuracy for the various models trained using the error function prediction.

To validate the usefulness of $g_n(X_i, \hat{Y}_i)$ for improving training, we train the Deep Salience vocal pitch model Bittner et al. (2017) three times[7] using three different training sets. The training sets are subsets of DALI, and contain the $\hat{Y}_i$ of all the songs that have a Normalized Cross-Correlation $> .9$ Meseguer-Brocal et al. (2018). This results in a training set of 1837 songs. The three sets are defined as follows:

1. **All** data. Trained using all time frames, D (Eq. 6.8).

2. **Filtered** data. Trained using the filtered, "non-error" time frames, F (Eq. 6.9), where the output of g has been binarized with a threshold of 0.5. With this data we tell the model to skip all the noisy labels.

3. **Weighted** data. Trained using all time frames, D, but during training, the loss for each sample is weighted by $1 - g_n(X_i, \hat{Y}_i)$. This scales the contribution of each data point in the loss function according to how likely it is to be correct.

We test the performance of each model on two polyphonic music datasets that contain vocal fundamental frequency annotations: 61 full tracks with vocal annotations from MedleyDB Bittner et al. (2014) and 252 30-second excerpts from iKala Chan et al. (2015). We compute the the generalized[8] Overall Accuracy (OA) which measures the percentage of correctly estimated frames, and the Raw Pitch Accuracy (RPA) which measures the percentage of correctly estimated frames where a pitch is present, which are standard metrics for this task Bittner and Bosch (2019); Salamon et al. (2014). The distribution of scores for each dataset and metric are shown in Figure 6.7.

While the scale of the results are below the current state of the art Bittner et al. (2017). This result is likely expected due to the noisiness of DALI and the fact that vocal note transcription is a different task from fundamental frequency estimation. Despite this result, we see a clear positive impact on performance when data cleansing is applied (see Table 6.3). The overall trend we see is that training using filtered data outperforms the baseline of training using all the data with

[7]We train each new model from scratch, not using transfer learning

[8]using continuous voicing from the model output and binary voicing from the annotations

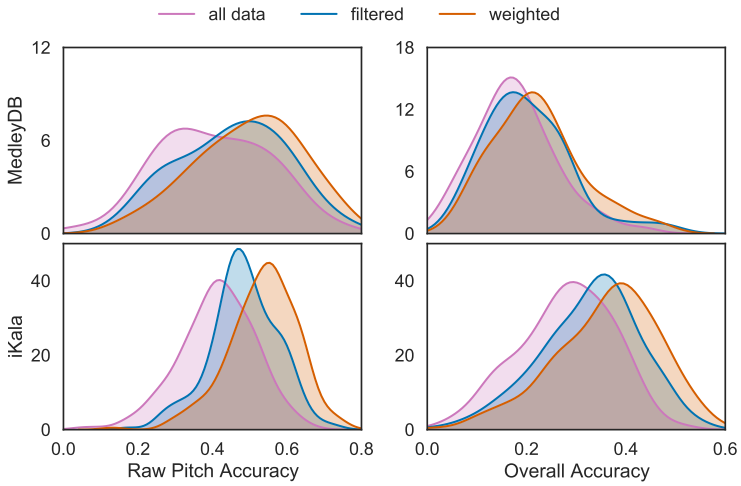


Figure 6.7: Distribution of scores for the three training conditions. Each condition is plotted in a different color. Scores for MedleyDB are shown in the top row and scores for iKala are in the bottom row. Raw pitch accuracy is shown in column 1 and overall accuracy is shown in column 2. The y-axis in all plots indicates the number of tracks.

statistical significance ($p < 0.001$ in a paired t-test) for all cases, indicating that our error detection model is successfully removing time frames which are detrimental to the model. We also see that, overall, training using all the data but using the error detection model to weigh samples according to their likelihood of being correct is even more beneficial than simply filtering. This suggests that the likelihoods produced by our error detection model are well-correlated with the occurrence of real errors in the data. These trends are more prominent for the iKala dataset than for the MedleyDB dataset – in particular, the difference between training on filtered vs. weighted data is statistically insignificant for MedleyDB while it is statistically significant ($p < 0.001$ in a paired t-test) for the iKala dataset. The iKala dataset has much higher proportion of *voiced* frames (frames with a pitch annotation) than MedleyDB. This suggests that the weighted data is beneficial for improving pitch accuracy, but does not bring any improvement over filtering for detecting whether a frame should have or do not have a pitch (voicing). Nevertheless, both conditions which used the error detection model to aid the training process see consistently improved results compared with the baseline.

Figure 6.8 illustrates the output of each of the three models for a track from the iKala dataset. We can see how the two versions that used the error probability function transcribe a melody closer to the expected output. Both models have considerably less dispersion with more coherent notes in time.

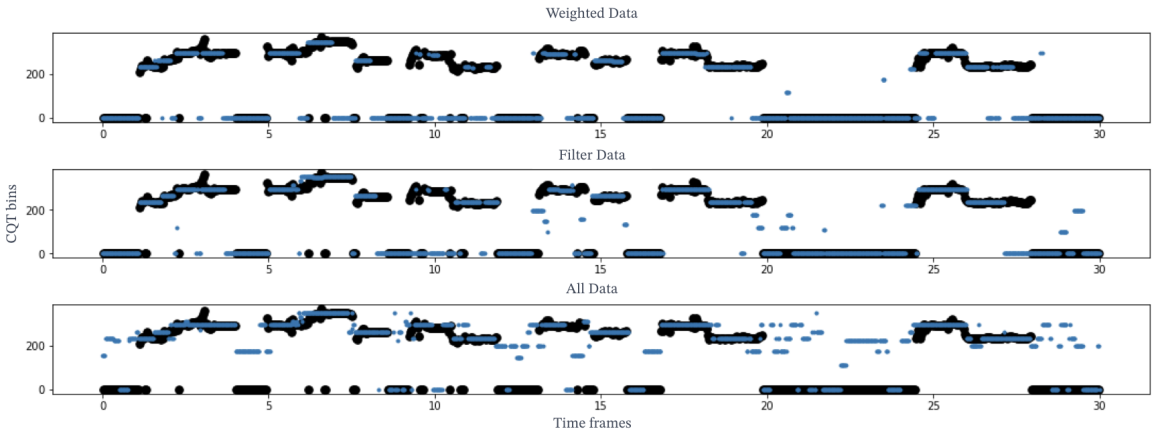


Figure 6.8: Example for each of the three models for an example track from iKala dataset. We can see how both the filtered and weighted versions output a transcription close to the expected one with less sparse output.

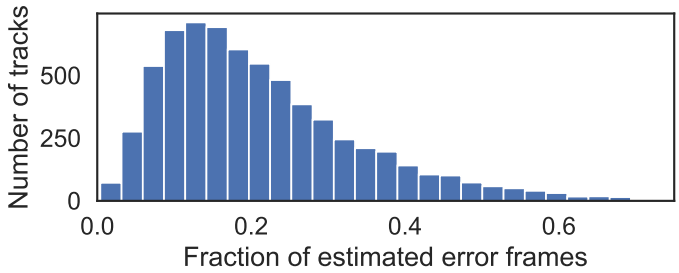


Figure 6.9: Histogram of the estimated error rate per track.

6.4.1 Estimated Quality of The DALI Dataset

As a final experiment, we ran the error detection model on the full DALI dataset (version 2) in order to estimate the prevalence of errors. We compute the percentage of frames per-track where the likelihood of being an error is ≥ 0.5. A histogram of the results is shown in Figure 6.9. We estimated that on average, 21.3% of the frames of a track in DALI will have an error in the note annotation, with a standard deviation of 12.7%. 31.1% of tracks in DALI have more than 25% errors, while 18.2% of tracks have less than 10% errors. We also measured the relationship between the percentage of estimated errors per track and the normalized cross correlation from the original DALI dataset Meseguer-Brocal et al. (2018), and found no clear correlation. This indicates that while the normalized cross correlation is a useful indication of the global alignment, it does not reliably capture the prevalence of local errors.

We manually inspected the tracks with a very high percentage of estimated errors ($> 70\%$) and found that all of them were the result of the annotation file being matched to the incorrect audio

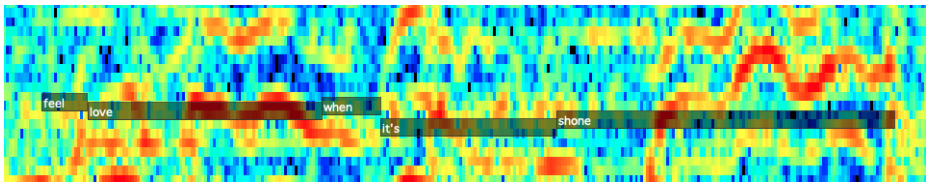


Figure 6.10: The CQT of an excerpt of the track in DALI with the lowest percentage error ($< 1\%$ error), with its annotations overlaid.

file (see Meseguer-Brocal et al. (2020b) for details on the matching process). On the other hand, we found that the tracks with very a low percentage of estimated errors ($< 1\%$) had qualitatively very high quality annotations. For example, Figure 6.10 shows an excerpt of the track with the lowest error rate along with a link to listen to the corresponding audio. While this is only qualitative evidence, it is an additional indicator that the scores produced by the error detection model are meaningful. The outputs of our model on DALI are made publicly available.

An error detection model that estimates the quality of a dataset can be used in several ways to improve the quality of a dataset. For one, we can use it to direct manual annotation efforts both to the most problematic tracks in a dataset, but also to specific incorrect instances within a track. Additionally, one of the major challenges regarding automatic annotation is knowing how well the automatic annotation is working. For example, the creation of the DALI dataset used an automatic method for aligning the annotations with the audio, and it was very difficult for the creators to evaluate the quality of the annotations for different variations of the method. This issue can now be overcome by using an error detection model to estimate the overall quality of the annotations for different variations of an automatic annotation method.

6.5 Discussion

We have introduced our data cleansing strategy for dealing with noisy labels.We showed that training directly a model that detects errors in the labels leads to a big improvement in downstream performance over training on the noisy data. In particular, the sample weights filter confidence outperforms when training with a filter version. Our approach is particularly interesting for complex prediction tasks with many multiple label classes since the binary problem is theoretically less complex than a model than the multi-class problem.

Our approach has some advantages over previous ones. For instance, it is simple and has a reduced architecture and training procedure. The output of the model can be used alongside existing datasets in a self-supervised way. Less effort is needed since we are using the dataset to mitigate the effects of noisy labels. It is especially useful for tasks that are complex and slow to train. Binary models can be simpler and trained much faster than models for the task itself. Another advantage is that our approach has a direct reflection on the data regardless of its final purpose, being independent

of it. This opens a wide range of possible usages. For instance, the error probability function gives a general idea of the current state of a dataset moving forward from the idea of datasets as ground truth, instead treating them as noisy estimates. We can also filter the dataset (globally e.g an audio track and locally e.g. a particular segment) depending on the sensitivity of the task, measuring things like *how much noisy points affect this particular task?*. It can be also seen as a way of evaluating the accuracy of a model in unlabeled data given a direct sense of the performance in the real world scenario. It can be used in a teacher-student paradigm schema gathering automatically training data from the Web and, instead of using all label provided by the teacher as the truth, they can be filtered with our error detection function, removing the noisy labels. Finally, in the context of **active learning**, the error probability function can select the noisiest data points for manually annotations or measure a human annotator's reliability.

6.5.1 Future work

Many things remain for future works. We plan to use our error detection models as a guide for testing the effectiveness of automatic label correction techniques described in Chapter 5.2. We can then automatically decide if the new labels are better or worse than the previous one.

Formulating the problem as a standalone binary supervised classification problem requires labeled data. Given a noisy dataset, the first step is to define the dataset $\mathcal{S}_{I^+} = \{((x_i, \hat{y}_i), z_i)\}_{i \in I}$ to train our error detection model. We have explored an initial hypothesis where the true labels are found comparing $\hat{y}_i$ with the estimated f_0 and the noisy labels are artificially created. Nonetheless, we can use now our error detection model to identify the true and noisy labels and define a new dataset $\mathcal{S}_{I^+} = \{((x_i, \hat{y}_i), z_i)\}_{i \in I}$. This is can be formulated again in a teacher-student paradigm where the first error detection model is the teacher that labels the dataset used to train a new error detection model.

We also plan to extend our approach to multiple domains with similar label noise such as objects positioned in an image or speech aligned with the text.

6.6 Conclusions

In this chapter, we have introduced our novel data cleansing technique that automatically locates errors in the noisy labels of DALI. We have shown how this technique can be effectively used to train a downstream task, improving the performance of same architecture by almost 10%.

Since the goal of this thesis is not to create a perfect dataset but rather to explore the interaction between audio and lyrics, we did not pursue further in this direction.

With this chapter, we close the work done concerning DALI, a big multimodal dataset with lyrics aligned in time. We list our main contributions as:

1. we created a multimodal dataset with lyrics aligned in time at different levels of granularity with different types of audio (original and multi-tracks) leveraging data from the Web, using novel techniques such as the teacher-student paradigm and a methodology where model creation and data curation help each other.

2. we showed that the teacher-student paradigm is a good solution while dealing with unlabeled to improve the performance of models such as our SVD.

3. we provided different proxies such as the NCC and the f_0 correlation to get to know the quality of the labels.

4. we explored the CTC internal alignment and several configurations to overcome the local noisy labels still present in the dataset.

5. we have directly addressed the noisy label issue by developing a novel data cleansing method for automatically evaluating the quality of the labels. This method is especially interesting for position-dependent multiclass labels with non-defined errors nor class dependencies.

In the following chapters, we use DALI for exploring direct interaction between lyrics and audio for MIR tasks.

Chapter 7

Improving lyrics segmentation combining text and audio

The first multimodal case scenario that we addressed with DALI is lyrics segmentation. The goal of this task is to detect the boundaries between sections (e.g. chorus, verse) of a song. That is, given a song composed by k lyrics lines $X = \{x_1, x_2, ..., x_k\}$ we want to find the lines $x_i \in X$ that define the section borders. The task can be easily formulated in a supervised learning way as a binary classifier $p(y_i|x_i)$ with $y_i = 1$ when x_i defines a boundary block and $y_i = 0$ otherwise. In DALI, each text line $a_i \in X$ is associated with a concrete audio section. We hypothesize that the text and audio modalities should capture complementary structure. In this chapter, we show how a **multimodal** representation composed by audio and text performs significantly better than using text information only.

7.1 Segmenting lyrics using text information

Understanding the structure of the lyrics is an important task for music content analysis (Cheng et al., 2009b; Watanabe et al., 2016). It permitts splitting a song into semantically meaningful sections, enabling a description per section (rather than a global description of the whole song). The detection of the lyric structures can be made in two steps. We first segment in sections, using the repeated patterns presented in the lyrics and then, we label each section (e.g. intro, verse, chorus). To better understand the problem, consider the example illustrated in Figure 7.1 where we show the lyrics and the corresponding segmentation into sections. Horizontal green lines indicate the section boundaries we aim to find. Even though it is frequently assumed that boundaries correspond to line breaks, this does not always hold. Additionally, line breaks are usually annotated by users and they are not necessarily identical to the intended segmentation defined by the songwriter. This

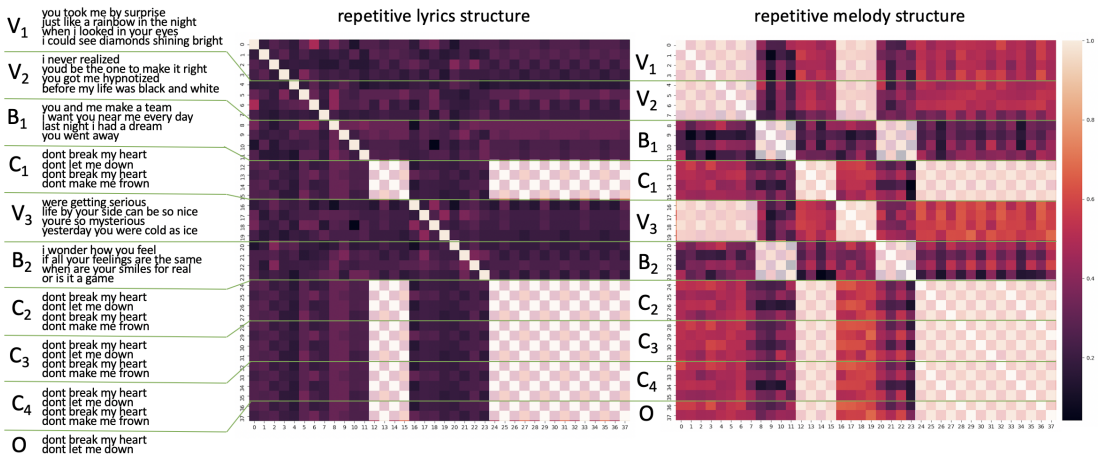


Figure 7.1: Lyrics (left) of "Don't Break My Heart" by Den Harrow. We illustrated two different SSM based on lyrics (middle), and on audio (right). Lyrics section boundaries (green lines) coincide with highlighted rectangles in lyrics and melody patterns in the SSM. Reprinted from (Fell et al., 2018)

motivated researchers to develop methods to automatically segment lyrics. This task is similar to music structure detection, where we automatically estimate the temporal structure of a music track by analyzing the repetitive audio patterns.

It is habitual to use textual features (e.g., n-grams and characters count), which have been proven effective (Fell et al., 2018; Mahedero et al., 2005; Watanabe et al., 2016; Baratè et al., 2013). The first attempt focused on lyrics with a recognizable structure, identified using relevant heuristics such as the length of each line and the total lenght of the lyrics or the relative position of a section border in the song (Mahedero et al., 2005). They tested their algorithm on a small dataset of 30 lyrics, 6 for each language (English, French, German, Spanish and Italian), which had previously been manually segmented. Authors have also looked for recurrent and non-recurrent groups of lines with a rule-based method to label the different sections (Baratè et al., 2013). Lyrics segmentation as a binary classification task was introduced as a solution to model repeated patterns (Watanabe et al., 2016). Given a song composed by k lyrics lines $X = \{x_1, x_2, ..., x_k\}$ this approach looks for $p(y_i|x_i)$ with $y_i = 1$ when x_i defines a boundary between sections and $y_i = 0$ otherwise. They use as features a Self-Similarity Matrices (SSM) (Foote, 2000), a feature quite popular in MIR that highlights distinct patterns, revealing the underlying structure. Given a song described as a set of sequential elements $X = \{x_1, x_2, ..., x_k\}$, SSM is the result of computing a similarity measure between the two corresponding elements $SSM_{i,j} = similarity(x_i, x_j)$ for all the possible combinations of the set X. As a result, each element is compared with all the elements in the set, creating a matrix that directly highlights similar elements. Diagonals (and sub-diagonals) and blocks (or the absence of them) are the two main patterns in a SSM. While diagonals indicate sequences that are repeated,

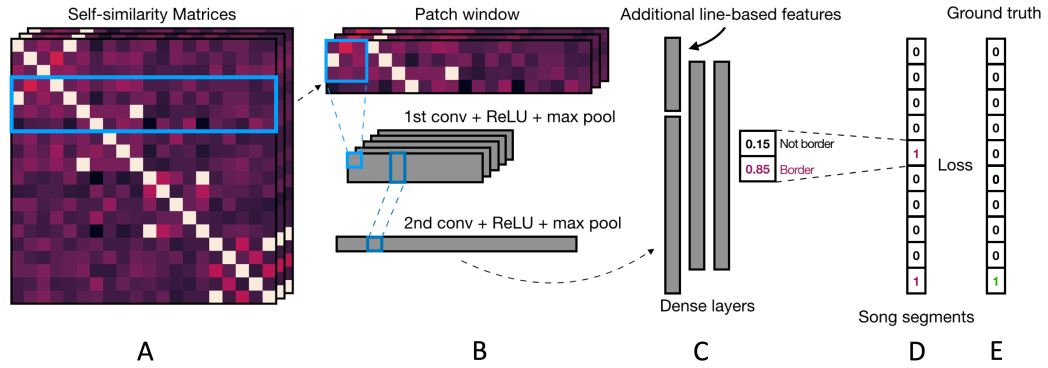


Figure 7.2: CNN model for inferring lyrics segmentation. Reprinted from (Fell et al., 2018)

blocks indicate sequences in which all the elements are highly similar to each another. Both patterns are indicators of sections. Figure 7.1 highlights two examples of SSM with the repetitive structure of the lyrics. Despite being a local element, each row/column of a SSM captures the similarity with all the other elements in the song.

This chapter extends previous work done in Fell et al. (2018). Authors use the binary formalization (Watanabe et al., 2016) with a CNN to detect section boundaries by leveraging the repeated text patterns conveyed in a SSM (see Figure 7.2). The model predicts if a lyrics line y_i is a border $p(y_i|x_i) = 1$ or not $p(y_i|x_i) = 0$ evaluating the corresponding SSM row plus some additional context rows around x_i. It has two convolutional layers each one with max-pooling after to downsample each feature to a scalar. The resulting feature vector is concatenated with the line-based features and is used as input to a series of fully-connected layers. The last layer has a single neuron with a *sigmoid* activation. The model produces the final probability using a binary cross-entropy loss between the prediction and the ground truth label. Authors use three different SSM respectively based on different line-based text similarity measures on characters, phonetic information or syntax characteristics. The best results are obtained with the **string similarity (`str`):** a normalized Levenshtein string edit similarity between the characters of two text lines (Levenshtein, 1966) used also in (Watanabe et al., 2016).

However, this approach fails where there is no clear structure in the lyrics, for instance when sentences are never repeated or in the opposite case when they are always repeated. For instance as we can see in Figure 7.1, the verses (V_i) and bridges (B_i) have not repetive pattern in the SSM extracted from the lyrics (middle). The reason is that these verses and bridges have different lyrics. However and since these lines share the same melody, we can easily see how these patterns arise in the SSM extracted from the audio (right) using *IrcamDescriptors* (Peeters and Rodet, 2004). These are the cases we target in this chapter. We hypothesize that since melodies are often repeated, the part of the structure which is not captured in the text may arise from the audio.

7.2 Formalization

Following the formalization described at Chapter 1.1, we use the multimodal information as complementary to improve lyrics segmentation:

How is the multimodal model constructed? We opt for a **model-agnostic** multimodal learning method that re-uses the architecture of a model proven to perform well for this task when used in one domain.

What multimodal information? We aim to construct a joint representation between audio and text where each domain captures complementary structure information. In the pre-processing step, both domains are treated independently to obtain a coordinate representation, the SSMs. This input is fed to the model that processes it and finds a joint space to infer the boundaries.

When is the context information used? This is a traditional early-fusion approach where the two domains are merged as input to the model. Since the baseline model is a CNN, we just simply concatenate both representations as extra channel inputs. The model is then in charge of processing it and finds the needed relationships between domains.

7.3 Multimodal version using text and audio information

7.3.1 Dataset

We use the first version of DALI with 5358 songs. We focus only on the time and text information of lines $A_{lines} = (a_{k,lines})_{k=1}^{K_{lines}}$ where $a_{k,lines} = (t_k^0, t_k^1, l_k)_{lines}$ (see Chapter 3.2). Since we know which lines are in each paragraph we can easily define the section boundary lines, $(a_i, y_i) = 1$ when the line is the last of a paragraph (see Figure 7.3a). To be consistent with previous approaches (Watanabe et al., 2016; Fell et al., 2018), we only select songs that have at least 5 sections reducing the number of songs to 4784.

As described earlier (see chapter 4.1), the raw lines extracted from the karaoke resources have been merged to create the paragraph level using as reference the paragraphs in WASABI. This process may induce errors. For this reason, we compute a new proxy that indicates the accuracy of the merging process. Let $D = D_0, D_1, ..., D_u$ be the paragraphs in DALI and $W = W_0, W_1, ..., W_v$ the paragraphs in WASABI, we can compute the similarity between them using **String similarity** `str`:

$$merge(D, W) = \min_{0 \leq i \leq u} \{ \max_{0 \leq j \leq v} \{ \text{str}(D_i, W_j) \} \} \tag{7.1}$$

This metric finds the paragraph in DALI with the lowest string similarity to a paragraph in WASABI. Note how this is quite restrictive because it only focuses on the worst single paragraph

```
i love you                          |                    y=0
is all that you can say             |                    y=0
years gone by and still             |— section 1 ->      y=0
words dont come easily              |                    y=0
like i love you i love you          |                    y=1
————————————————————————————————————
but you can say baby                |                    y=0
baby can i hold you tonight         |                    y=0
baby if i told you the right words  |— section 2 ->      y=0
oooh at the right time              |                    y=0
youll be mine                       |                    y=1
————————————————————————————————————
```

(a) Example of how the target values y_i are assigned to each line.

Version	Merge value	Number of songs
full dataset	-	4784
M^+	high (90-100%)	1048
M^0	med (52-90%)	1868
M^-	low (0-52%)	1868

(b) The DALI dataset partitioned by $merge(D, W)$

in DALI and does not take into account the rest. We partition DALI into three different subsets based on the different values of *merge* (see Table 7.3b). The M^+ subset consists of 50842 lines and 7985 section boundaries. It has 72% of the lines in English, 11% in German, 4% in French, 3% in Spanish, 3% in Dutch and 7% in other languages.

7.3.2 Self Similarities

Each $a_{k,l} = (t_k^0, t_k^1, l_k, i_k)_{lines}$ connects a text line l_k with a particular audio segment (t_k^0, t_k^1). We can then add to the text SSMs the new SSMs that compare audio segments. The audio SSMs captures complementary information such as melodic or harmonic structures that complement the text structures. Since the audio SSMs have also the same dimensions, they can be simply stacked as an extra channel to the original input. The architecture of the model stays the same. Instead of comparing the raw audio of each x_i itself, we extract two sets of well-known audio features: the **Mel-frequency cepstral coefficients** ($mfcc \in \mathbb{R}^{14}$) (Davis and Mermelstein, 1980), to emphasize the part of the signal that is related (with our understanding) to the musical timbre and the **Chroma** ($chroma \in \mathbb{R}^{12}$) (Fujishima, 1999) to describe the harmonic information of each frame, computing the "presence" of the twelve different pitch-classes.

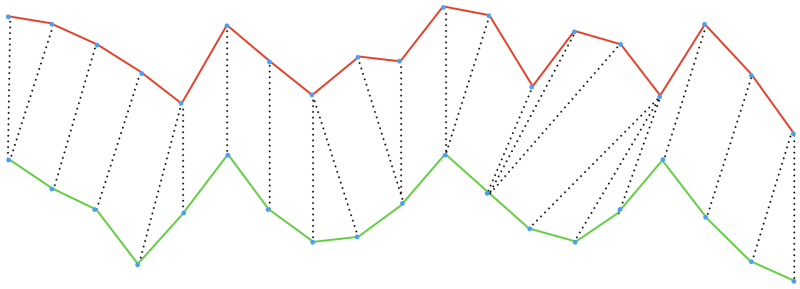


Figure 7.4: Illustrative example of the DTW alignment between two sequences.

We then compute the audio SSM by measuring the similarities between all the audio segments $X = \{x_1, x_2, ..., x_k\}$ defined by the $a_{k,l}$. Considering that each audio segment may have a different length, we use as similarity the last element of the DTW cost matrix. Given two lines i and j

of length p and q respectively described as a sequence of features $x_{i,p} = \{x_{i,0}, x_{i,1}, ..., x_{i,p}\}$ and $x_{j,q} = \{x_{j,0}, x_{j,1}, ..., x_{j,q}\}$, DTW (Sakoe and Chiba, 1978) applies a nonlinear warping to best align their similar areas (even if they are out of phase in the time axis) with a minimal distance between them (see Figure 7.4). DTW is formulated as an optimization problem:

$$DTW(x_{i,p}, x_{j,q}) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_{i,p}, x_{j,q})} \tag{7.2}$$

where $d(x_{i,p}, x_{j,q})$ is the distance between the feature frames i and j of x and y respectively, and $\pi = [\pi_0, \ldots, \pi_K]$ the alignment path between $x_{i,p}$ and $x_{j,q}$ where each element is a pairs $\pi_k = (i_k, j_k)$ with $0 \leq i_k < n$ and $0 \leq j_k < m$. The path has to be **monotonic** (it does not go backwards in time), **contiguous** and covers full sequences, $\pi_0 = (0, 0)$ and $\pi_K = (n-1, m-1)$.

The algorithmic solution to this problem can be found efficiently using *dynamic programming*. The algorithm creates a cumulative matrix where each point cumulative the minimum distances of the adjacent elements.

$$DTW_{cost}(x_{i,a}, x_{j,b}) = d(x_{i,a}, x_{j,b}) + min \left\{ \begin{array}{c} DTW_{cost}(x_{i,a-1}, x_{j,b}) \\ DTW_{cost}(x_{i,a-1}, x_{j,b-1}) \\ DTW_{cost}(x_{i,a}, x_{j,b-1}) \end{array} \right\} \quad ^{1} \tag{7.3}$$

Once DTW_{cost} is computed, we obtain the alignment path by backtracking from the last position $(p-1, q-1)$ to the origin $(0, 0)$ with the minimum cost. The most significant advantage of DTW is its invariance against shifting and scaling in the time axis and its capacity to align both signals where their corresponding similar areas are linked.

We use as distance between features $d(x_{i,a}, x_{j,b}) = 1 - |cosine(x_{i,a}, x_{j,b})|^2$ which is a normalized value between 0 (no distance between features) and 1 (very different features) and does not depend on the dimensionality of the features. Because of this normalization, the cost matrix stores an accumulation where each feature frame contributes equally (a normalized value between 0 and 1). Thus, by simply dividing by the length of the path to get to that point we compute a normalized cost. This allows us to use as similarity to use $s(x_{i,p}, x_{j,q}) = 1 - (DTW_{cost}(p, q)/l)$ where l is the lenght of the best alignment path between both signals (the minimal path of the cost matrix) instead of $s(x_{i,p}, x_{j,q}) = 1/DTW_{cost}(p, q)$ which depends on signal lenghts. The final $s(x_{i,p}, x_{j,q})$ has always a value between 0 and 1, which corresponds to the mean cost contribution to the final path per feature frame. Thereby, the different $DTWs_{cost}$ that compare the audio segments $X = \{x_1, x_2, ..., x_k\}$ has the same range of values allowing us to direclty compute the audio SSM (see Figure 7.5). We employ two SSM matrices ssm_{mfcc} and ssm_{chroma}.

[1] Researchers have explored many other constrains to build this matrix such as *slope weighting* by a factor $w(a-1, b), (a-1, b-1), w(a, b-1)$ or proposing different *step patterns* like $(a-1, b-1), (a-1, b-2), (a-1, b-1)$. These variations can favor different alignment behaviors.

[2] The *cosine* of 0° is 1 and less than 1 for any angle in the interval $(0, \pi]$

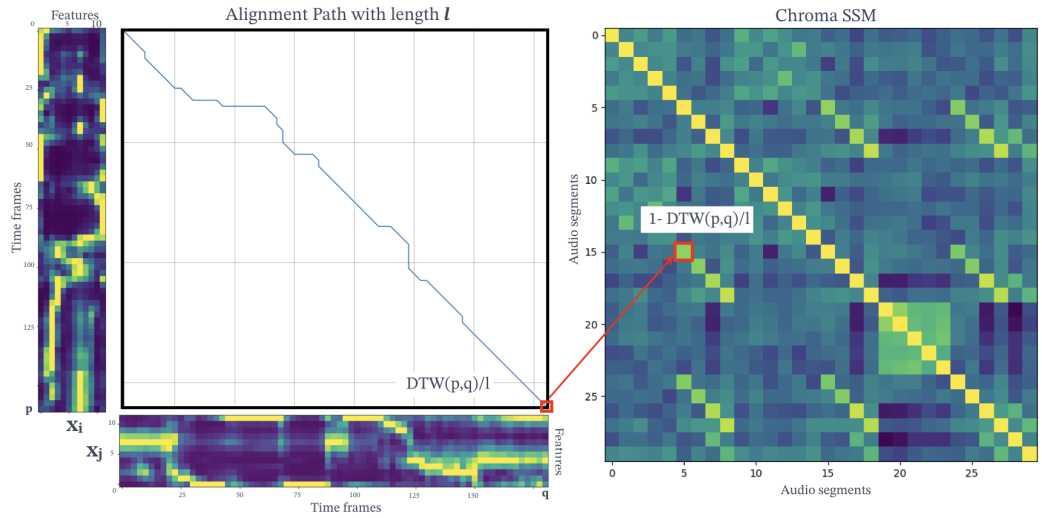


Figure 7.5: Audio SSM computation. [Left] The DTW computes the alignment between two audio features sequences x_i and x_j of lengths p and q. We use the inverse of the global alignment cost as a similarity measurement between audio sequences of different lengths. [Right] Repeating this procedure and comparing all the audio segments $X = \{x_1, x_2, ..., x_k\}$ in a song defined by $a_{k,l}$ we create the final SSM.

7.4 Evaluation of the multimodal lyrics segment detector

We compare different versions of the original model (Fell et al., 2018) (see Figure 7.2). Each one is trained with different input data: *text*-based , *audio*-based, and *multi*modal-based with both text and audio features. We use as dataset the M^+ subset. We split the data randomly into training and test sets using a 5-fold cross-validations. The cross validation is performed twice every time with different random initialization. The results are depicted in Table 7.1.

If we focus on the F_1, the model $text_{str}$ trained on M^+ performs similary (70.8%) to the *audio* one (70.4% for $audio_{mfcc,chroma}$). However, each indivual features perform worse with 65.3% for

Model	*Features*	*P*	*R*	F_1
text	{str}	78.7	64.2	70.8
audio	{mfcc}	79.3	55.9	65.3
	{chroma}	76.8	54.7	63.9
	{mfcc, chroma}	79.2	63.8	70.4
multi	{str, mfcc}	80.6	**69.0**	73.8
	{str, chroma}	82.5	**69.0**	74.5
	{str, mfcc, chroma}	**82.7**	**70.3**	**75.3**

Table 7.1: Results with multimodal lyrics lines on the M^+ dataset in terms of Precision (P), Recall (R) and F-measure(F_1) in %. Note that the $text_{str}$ model is the same configuration as in Fell et al. (2018), but trained on different dataset.

Dataset	Model	Features	P	R	F_1
M^+	*text*	{str}	78.7	64.2	70.8
	audio	{mfcc, chroma}	79.2	63.8	70.4
	multi	{str, mfcc, chroma}	82.7	70.3	75.3
M^0	*text*	{str}	73.6	54.5	62.8
	audio	{mfcc, chroma}	74.9	48.9	59.5
	multi	{str, mfcc, chroma}	75.8	59.4	66.5
M^-	*text*	{str}	67.5	30.9	41.9
	audio	{mfcc, chroma}	66.1	24.7	36.1
	multi	{str, mfcc, chroma}	68.0	35.8	46.7

Table 7.2: Results with multimodal lyrics lines for the alignment quality ablation test on the datasets M^+, M^0, M^- in terms of Precision (P), Recall (R) and F-measure(F_1) in %.

$audio_{mfcc}$ and 63.9% for $audio_{chroma}$. As the $mfcc$ feature models timbre and instrumentation, whilst the chroma feature models melody and harmony. They provide complementary information that benefits the model with both features, $audio_{mfcc,chroma}$.

Most importantly, the overall best performing model is a combination of the *text* and *audio* features. It achieves the best results in all three evaluation metrics: *precision*, *recall*, and a F_1 of 75.3%. This shows that text and audio modalities capture complementary structures that are beneficial for the lyrics segmentation. Additionally, each *multi* version also outpeforms the *text* and *audio* models. $multi_{str,mfcc}$ and $multi_{str,chroma}$ achieve a performance of 73.8% and 74.5%.

In Table 7.2, we give the performances on the three subsets of DALI, M^+, M^0 and M^- using the feature that performed best on the M^+ i.e. $text_{str}$, $audio_{mfcc,chroma}$, and $multi_{str,mfcc,chroma}$. Each subset follows its own 5-fold cross-validations procedure. We find that independent of the modality (text, audio, mult.), all models perform significantly better when the merge value is higher. Moreover, even if the results decrease, the *multi* version always performs better than the one domain versions.

7.5 Conclusions

In this chapter, we used DALI for the first time to address the task of lyrics segmentation on synchronized text-audio representations. Since each song contains the lyrics aligned with the audio, we can compute several SSMs for the two domains. We showed that exploiting both textual and audio-based features outperforms the systems that rely on purely text-based features proving that each domain captures complementary structures that benefit to the overall performance. This chapter is the result of a collaboration with Michael Fell and Elena Cabrio who trained and tested the models.

Chapter 8

Conditioned U-Net

Data-driven models for audio source separation such as U-Net (see Chapter 2.1.5) or Wave-U-Net are usually models separated to and specifically trained for a single task, e.g. a particular source separation. Training them for various tasks at once commonly results in worse performances than training them for a single specialized task. In this chapter, we introduce the Conditioned-U-Net (C-U-Net) which adds a control mechanism to the standard U-Net. The control mechanism allows us to train a unique and generic U-Net to perform the separation of various sources. The C-U-Net decides the source to isolate according to a one-hot-encoding input vector. The input vector is embedded to obtain the parameters that control Feature-wise Linear Modulation (FiLM) layers. FiLM layers modify the U-Net feature maps in order to separate the desired source via affine transformations. The C-U-Net performs different source separations, all with a single model achieving the same performances as the separated ones at a lower cost. The multitask formalization also serves as an exploration of how FiLM can effectively control the behavior of a generic U-Net. This idea will be extended in the next chapter for improving vocal source separation using the DALI.

8.1 Introduction

Generally, in MIR we develop separated systems for specific tasks. Facing new (but similar) tasks require the development of new (but similar) specific systems. This is the case of data-driven music source separation systems.

Music is usually distributed as mono channel or 2-channel stereo (left and right) where all the instruments are mixed. However, there are many cases in which we are only interested in listening to or working with a single instrument or a particular combination of them. It serves also as an intermediary step for other MIR tasks such as fundamental frequency estimation or score transcription. The music source separation task aims to isolate the different instruments that appear in an audio mixture (a mixed music track), i.e. reversing the mixing process (Cano et al., 2018).

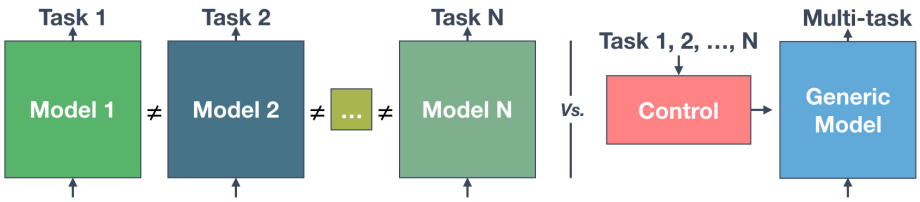


Figure 8.1: [Left part] Traditional approach: a separated U-Net is trained to separate a specific source. [Right part] Our proposition based on conditioned learning. The problem is divided in two: a standard U-Net (which provides generic source separation filters) and a control mechanism. This division allows the same model to deal with different tasks using the commonalities between them.

Source separation is one of the main MIR tasks. For decades, it has awakened the interest of researchers who have developed many approaches. Source position (to exploit the spatial position of the sources), kernel additive (to find local features presented in the spectrogram), non-negative matrix factorization (to factor the mix into a combination of basic sources activated at different time positions) or sinusoidal models (to approximate the mixture by several sinusoids) have been successfully applied (Rafii et al., 2018; Cano et al., 2018; Pardo et al., 2018). Currently and thanks to the increase of annotated datasets, data-driven methods have taken the lead. These methods use supervised learning where the mixture signals and the isolated instruments are available for training. The usual approach is to build separated models for each source to be isolated (Jansson et al., 2017; Stoller et al., 2018c). This has been proved to show great results. However, since isolating an instrument requires a specific system, we can easily run into problems such as scaling issues (100 instruments = 100 systems). Besides, these models do not use the commonalities between instruments. If we modify them to do various tasks at once i.e., adding fix numbers of output masks in the last layer, they reduce their performance (Stoller et al., 2018c).

Conditioned learning has appeared as a solution to problems that need the integration of multiple resources of information. Concretely, when we want to process one in the context of another i.e., modulating a system computation by the presence of external data. Conditioned learning divides problems into two elements: a **generic system** and a **control mechanism** that governs it according to external data. Although there is a large diversity of domains that use it, it has been developed mainly in the image processing field for tasks such as visual reasoning or style transfer. There, it has been proved very effective, improving the state of the art results (Perez and Wang, 2017; de Vries et al., 2017; Strub et al., 2018). This paradigm can be integrated into source separation, creating a generic model that adapts itself to isolate a particular instrument via a control mechanism. We believe that this paradigm can benefit to a great variety of MIR tasks such as multi-pitch estimation, music transcription or music generation.

8.2 Formalization

In this chapter, we study the application of conditioned learning for music source separation where an external context vector decides which instrument is to be separated. As described in Chapter 1.1, we can define our problem satisfying certain properties summarized as (Bengio et al., 2013):

How is the multimodal model constructed? Our system relies on a **standard U-Net system** not specialized in a specific task but rather in finding a set of generic source separation filters, that we **control** differently for isolating a particular instrument, as illustrated in Figure 8.1. The *conditioning* itself is performed using Feature-wise Linear Modulation (FiLM) layers (Perez and Wang, 2017), controlling the generic model model to perform different instrument source separations. Hence, this is a **model-based** multimodal learning approach.

What context information? Our system takes as input the spectrogram of the mixed audio signal and the control vector as context information for guiding the separation. It gives as output (only one) the separated instrument defined by the control vector. The main advantages of our approach are - direct use of commonalities between different instruments, - a constant number of parameters no matter how many instruments the system is dealing with - and scalable architecture, in the sense that new instruments can be potentially added without training from scratch a new system.

When is the context information used? We insert these layers in the encoder of the *generic* network for creating different latent spaces concerning the context information that defines the instrument to separate.

8.3 Related work

In this thesis, we focus only on the data-driven source separation approaches. We review only works related to this approach as well as conditioning in audio and multitask source separation. Note that many of the reviewed papers have been published after carrying out the work presented in this chapter.

Source separation based on supervised learning

Over the past decade, researchers have developed a wide variety of source separation methods for many different scenarios. We refer the reader to Rafii et al. (2018); Cano et al. (2018); Pardo et al. (2018) for an extensive and comprehensive overview. In data-driven approaches, we have access to the mixture and the isolated sources that compose it. Thereby, models learn in a supervised learning way to either compute a mask to isolate the source from the background (for instance Ideal Binary/Ratio Mask or Spectral Magnitude Mask) or to obtain directly the clean spectral representations (e.g. magnitude/power spectrum or mel spectrum) (Wang and Chen, 2018). The appearance of large and

representative datasets (The Music Audio Signal Separation (MASS) dataset (Vinyes, 2008), the Musdb dataset (Rafii et al., 2017) for the Signal Separation Evaluation Campaigns (SiSEC) (Vincent et al., 2009) or MedleyDB (Bittner et al., 2014) among others) has accelerated the progress and boosted the separation performance. Nowadays, neural networks have taken the lead.

The first approaches rely on global features across the entire frequency spectrum, over a longer period. Wang et al. (2014) proposes to separate speech signals from a noisy mixture, estimating the Ideal Ratio Mask (IRM) and the short-time Fourier transform Spectral Mask (FFT-MASK) with a fully-connected network. Huang et al. (2014); Huang et al. (2015) implement a method for monaural source separation for speech separation, singing voice separation, and speech denoising. They use a Recurrent Neural Network (RNN) to model the temporal evolution of timbre features for each source, taking as inputs single frames of the magnitude spectrogram of a mixture. Timbre features are used to compute a soft masking function that isolates the desired source. Nugraha et al. (2016a,b) work with multichannel source separation, using both phase and magnitude information. They use a fully-connected architecture to estimate the magnitude spectra of the sources combined with spatial covariance matrices to encode the spatial characteristics and a final multichannel Wiener filter. Uhlich et al. (2015) also implement a fully-connected architecture where the input concatenates multiple frames of the magnitude spectrogram of a mixture, modeling timbre features across multiple time frames. While these approaches work well, they do not completely exploit local time-frequency features.

The encoder-decoder architecture together with CNN has become one of the most popular choices. Chandna et al. (2017) estimates time-frequency soft masks that are applied to the magnitude spectrogram. The encoder consist of a 'vertical' CNN layer to capture local timbre information followed by a 'horizontal' CNN layer to capture temporal evolution. The output is connected to a fully-connected layer to perform a dimensional reduction. The decoder has a fully-connected layer with as many neurons as the size of the 'horizontal' CNN and a sucession of deconvolution layers to inverse the convolution that compute the final soft masks. Jansson et al. (2017) adapts the U-Net architecture to separate the vocal and accompaniment components, training a specific model for each task. It consists of an encoder-decoder with residual/skip connection between blocks at similar depths. The architecture is described in detail in the following sections. The two previous approaches use the phase of the original mixture to recompute the waveform from the magnitude spectrogram. Uhlich et al. (2017) computes also masking on the magnitude spectrogram, using a bi-directional *LSTM* model after compressing the frequency and channel information with a fully-connected block. The core of the model are three *LSTM* layers with a residual/skip connection between the input to the first *LSTM* and the output of the last one. The output is decoded back to its original input dimensionality with two fully-connected blocks, computing the final mask. A post-processing step with a multichannel Wiener filter obtains the final waveform. This work has been open-source implemented in Open-Unmix (Stoter et al., 2019). Wang et al. (2018) propose an iterative phase reconstruction

procedure where the STFT and its inverse are replaced by trainable linear convolutional layers.

Since the output of these works is the magnitude spectrogram, they need to reconstruct the audio signal using a phase that is not the original one. This potentially leads to artifacts. Wave-U-Net proposes to apply an architecture similar to the U-Net one but on the audio-waveform (Stoller et al., 2018c). They also adapt their model for isolating different sources at once by adding as many outputs as sources to separate. However, this multi-instruments version performs worse than the separated one (for vocal isolation) and has to be retrained to different source combinations. Défossez et al. (2019) add to the Wave-U-Net architecture a two layers bi-directional *LSTM* in-between the encoder and the decoder and introduce a $1x1$ convolution after each convolution block to halve the number of channels. They also propose a semi-supervised approach to leverage unlabeled multitrack songs. Conv-TasNet (Luo and Mesgarani, 2019) is also a time-domain approach that adds a separator to the encoder-decoder architecture. Once the encoder computes a latent representation, the separator learns a mask in this latent space. The separator is based on a temporal convolutional network (TCN) (Lea et al., 2016) with 8 stacked dilated convolutional blocks with exponentially increasing dilation factors. (Chandna et al., 2019) propose a different paradigm. Instead of extracting the vocals from the mixture, they use a convolutional network with residual/skip connection and dilated convolutions to estimate the parameters of a synthesizer which synthesizes the vocal track, without any interference from the backing track. Finally, the separation can be improved using a generative adversarial training process (Stoller et al., 2018a).

8.3.1 Conditioning in Audio

Conditioning has been mainly explored in **speech generation**. In the WaveNet approach (van den Oord et al., 2016, 2017) the speaker identity is fed to a conditional distribution adding a learnable bias to the gated activation units. A WaveNet modified version is presented in (Shen et al., 2018). The time-domain waveform generation is conditioned by a sequence of Mel spectrogram computed from an input character sequence (using a recurrent sequence-to-sequence network with attention). In **speech recognition** conditions are used in (Kim et al., 2017), applying conditional normalisation to a deep bidirectional LSTM (Long Short Term Memory) for dynamically generating the parameters in the normalisation layer. This model adapts itself to different acoustic scenarios. In (Kim et al., 2017), the conditions do not come from any external source but rather from utterance information of the model itself. They have been also used in **music generation** for accompaniments conditioned on melodies (Huang et al., 2018) or incorporating history information (melody and chords) from previous measures in a generative adversarial network (GAN) (Yang et al., 2017). It has been also proved to be very efficient for **piano transcription** (Hawthorne et al., 2018): the pitch onset detection is internally concatenated to the frame-wise pitch prediction controlling if a new pitch starts or not. Both, onset detection and frame-wise prediction are trained together.

Multitask and conditioning source separation

Early work study informed source separation in an encoding/decoding framework (Liutkus et al., 2012; Parvaix and Girin, 2010; Parvaix et al., 2009). In the encoding stage, it is assumed to have access to the isolated sources to extract characteristic descriptors (parameters that describe the structure of the source signals, or their contribution to the mixture) for each source. This extra information is imperceptibly embedded within the mixture, using a watermarking technique. The decoding stage has as input the new 'watermarked' mixture (with the descriptors of each source) and extracts the characteristic descriptors, using them to perform the final source separation. Note how this framework has two distinct and independent processes that differ from the encoder/decoder architecture used in deep neural network architectures. This approach has been studied together with molecular grouping (gather of close Modified Discrete Cosine Transform coefficients) (Parvaix et al., 2009; Parvaix and Girin, 2010) or modeling the sources as independent and locally stationary gaussian process, and the mixture as linear filter (Liutkus et al., 2012).

Slizovskaia et al. (2019) extend the Wave-U-Net architecture to perform a non-fixed number of separations. Unlike previous works that isolate popular music instruments (vocals, drums or bass), they concentrate on classical instruments (violin or viola). They add an external model that computes the necessary parameter to perform a multiplicative condition in the bottleneck (latent space) of the model. This block receives an additional condition vector that specifies the instrument to be separated. Kadandale et al. (2020) explore different loss strategies to allow the model to have a fixed number of multiple outputs. Kavalerov et al. (2019); Tzinis et al. (2019) aim to develop a universal sound separation, i.e. separating acoustic sources from an open domain, regardless of their class. They adopt a Conv-TasNet architecture and add a sound classifier that conditions the separation based on the semantic information extracted from the mixture. Assuming that it is known that the mixture has m sources (from a list of 527 categories) the classifiers predict the classes over time to condition the generic model. Lee et al. (2019) add a query-net to control the separation using a U-Net architecture. Given an audio query that defines the source to be isolated, query-net computes a latent representation that is both added as an input to the actual model that performs the separation and to condition the decoder part. Seetharaman et al. (2019) embed the mixture in gaussian spaces using *BLSTM* stack. A conditioning mask applied on the gaussian spaces is generated from a one-hot vector indicating the class as input. Meta-TasNet adds a parameter generator to the Conv-TasNet (Samuel et al., 2020). As before, a one-hot encoder codifies the desired instrument. The conditioning is performed in the latent space learning how to mask the important features. They apply a multi-stage architecture to iteratively upsampl low-resolution audio to high resolution.

Finally, there is a group of papers in which researchers, instead of finding generic source separation models independent of sources to isolate, investigate multitasking source separation by having the model jointly solve some additional (but related) task. They hypothesize that the insights learned

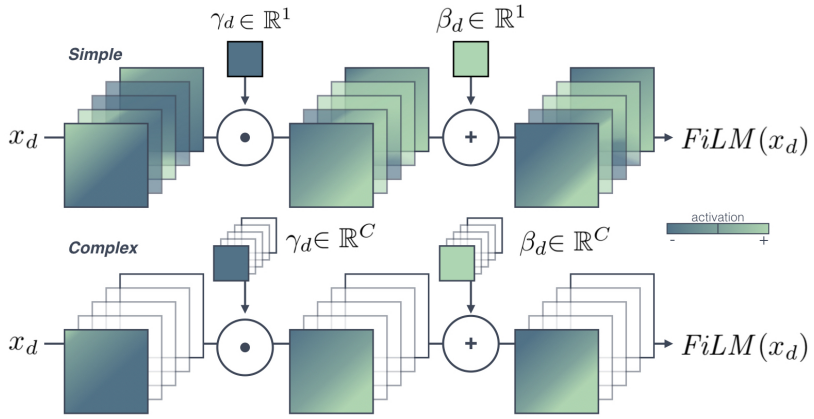


Figure 8.2: [Top] *FiLM simple* layer applies the same affine transformation to all the input feature maps x_d. [Bottom] In the *FiLM complex* layer, independent affine transformations are applied to each feature map C and a given depth of the net d.

for solving this extra task help in isolating a particular instrument. Most of the works focus on vocal separation. Early works such as (Durrieu et al., 2008) propose a two-step process where the vocal is isolated from the mixture before extracting the fundamental f_0. Nakano et al. (2019) invert the order and perform first f_0 extraction to enhance the vocal source separation. Possible errors in the f_0 extraction can be manually solved (Nakano et al., 2020). Jansson et al. (2019) explore many different configurations such as shared-encoder, cross-stitch, or stack operations. Stoller et al. (2018b) aims to extract vocal activity detection to stabilize and improve the performance of vocal separation. All these previous works used the U-Net as a base network. Finally, source separation is also quite related to music transcription (notating in a musical score). Manilow et al. (2019) simultaneously transcribe and separate multiple instruments, learning a shared musical representation for both tasks.

8.4 Conditioning learning methodology

8.4.1 Conditioning mechanism.

There are many ways to condition a network (see Dumoulin et al. (2018) for a wide overview) but most of them can be formalized as affine transformations denoted by the acronym FiLM (Feature-wise Linear Modulation) (Perez and Wang, 2017). FiLM permits to modulate any neural network architecture inserting one or several FiLM layers at any depth of the original model. A FiLM layer conditions the network computation by applying an affine transformation to intermediate features:

$$FiLM(x_d) = \gamma_d(z) \cdot x_d + \beta_d(z) \tag{8.1}$$

where $x_d \in \mathbb{R}^{W \times H \times C}$ is the input of the FiLM layer, the intermediate features of the **conditioned network**, at a particular depth d in the architecture, W and H represent the 'time' and 'frequency' dimension and C the number of feature channels (or feature maps) and (γ_d, β_d) and are parameters to be learned. They scale and shift x_d based on the external information, z. The output of a FiLM layer has the same dimension as the intermediate feature input x_d. FiLM layers can be inserted at any depth d in the controlled network.

As described in Figure 8.2, the original FiLM layer applies an independent affine transformation to each feature map (dimension C)[1]: $(\gamma_d, \beta_d) \in \mathbb{R}^C$ (Perez and Wang, 2017). γ_d and β_d are scalars applied to the feature map C of the input x at a given depth of the network d, needing as many parameters as input channels of features has. We call this a *FiLM complex* layer (**Co**). We propose a simpler version that applies the same $(\gamma_d, \beta_d) \in \mathbb{R}^1$ to all the feature maps (therefore γ_d and β_d do not depend on C). We call it a *FiLM simple* layer (**Si**). The *FiLM simple* layer decreases the degrees of freedom of the transformations to be carried out forcing them to be generic and less specialized. It also reduces drastically the number of parameters to be trained. As FiLM layers do not change the shape of x_d, FiLM is transparent and can be used in any particular architecture providing flexibility to the network by adding a control mechanism.

Even if this transformation (an affine operation with two parameters) is identical to batch normalization (see Chapter 2.1.6), it differs in some essential points. Batch normalization is fundamentally a feature standardization (zero mean and standard deviation of one) so that each feature has the same contribution. It additionally applies an affine transformation to modify its mean and variance in order to find the best values to take advantage of the non-linearity function. Therefore, the γ_d and β_d obtained do not depend on any external factor, but they serve to an optimization purpose. On the other hand, the γ_d and β_d computed for FiLM depend on the task at hand and they have different values for each task. Its main goal is to describe how to modulate the generic architecture, finding different specializations to carry on several tasks. Although they are inserted after batch normalization, this is not mandatory and they also apply to any type of feature (not necessarily standardized), showing similar results (Perez and Wang, 2017).

8.4.2 Conditioning architecture

A conditioning architecture has two components:

The conditioned network

It is the network that carries out the core computation and obtains the final output. It is usually a generic network that we want to behave differently according to external data. Its behavior is altered by the condition parameters, γ_d and β_d via FiLM layers.

[1] Or element-wise.

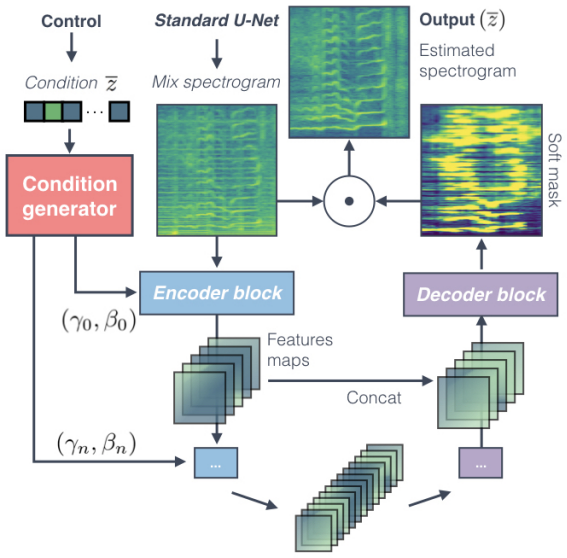


Figure 8.3: The C-U-Net has two distinct parts: the condition generator and a standard U-Net. The former codifies $\overline{z}$ (with the instrument to isolate) for getting the needed γ_d and β_d. The generic U-Net has as input the magnitude spectrum. It adapts its behaviour via FiLM layers inserted in the encoder part. The system outputs the desired instrument defined by $\overline{z}$.

The control mechanism - condition generator

It is the system that produces the parameters (γ_d's and β_d's) for the FiLM layers with respect to the external information z: the input conditions. It codifies the task at hand and provides the instructions to control the conditioned network. The condition generator can be trained jointly (Perez and Wang, 2017; Strub et al., 2018) or separately with the conditioned network (de Vries et al., 2017).

This paradigm clearly separates, from the main core computation, the task description and the control instructions.

8.5 Conditioned-U-Net for multitask source separation

We formalize source separation as a multi-tasks problem where one task corresponds to the isolation of one instrument. We assume that while the tasks are different they share many similarities, hence they will benefit from a conditioned architecture. We name our approach the **Conditioned-U-Net** (C-U-Net). It differs from the previous works where a separated model is trained for a single task (Jansson et al., 2017) or where it has a fixed number of outputs (Stoller et al., 2018c).

As in (Jansson et al., 2017; Stoller et al., 2018c), our **conditioned network** is a standard U-Net (see Chapter 2.1.5) that computes a set of generic source separation filters that we use to separate the various instruments. It adapts itself through the control mechanism (the condition generator)

with FiLM layers inserted at different depths. Our external data is a conditional vector $\overline{z}$ (a one-hot-encoding) which specifies the instrument to be separated. For example, $\overline{z} = [0, 1, 0, 0]$ corresponds to the drums. The vector $\overline{z}$ is the input to the control mechanism/condition generator that has to learn the best γ_d and β_d values such that, when they modify the feature maps (in the FiLM layers) the C-U-Net separates the indicated instrument i.e., it decides which features maps information is useful to get each instrument. The control mechanism/condition generator is itself a neural network that embeds $\overline{z}$ into the best γ_d and β_d. The conditioned network and the condition generator are trained jointly. A diagram is shown in Figure8.3.

Our C-U-Net can perform different instrument source separations as it alters its behavior depending on the value of the external condition vector $\overline{z}$. The inputs of our system are the mixture and the vector $\overline{z}$. There is only one output, which corresponds to the isolated instrument defined by $\overline{z}$. While training, the output corresponds to the desired isolated instrument that matches the $\overline{z}$ activation.

8.5.1 Conditioned network: U-Net architecture

We used the U-Net architecture proposed for vocal separation (Jansson et al., 2017), which is an adaptation of the U-Net for microscopic images (Ronneberger et al., 2015) (see Chapter 2.1.5). The U-Net follows an encoder-decoder architecture and adds residual/skip connections to it (see Chapter 2.1.6). The input and output are the normalized magnitude spectrograms of the DFT of the monophonic mixture and the instrument to isolate. DFT represents a signal as a sum of complex-valued Fourier coefficients (magnitude and phase) for sinusoids of varying frequency (see Figure 5.7).

Encoder

It creates a compressed and deep representation of the input by reducing its dimensionality while preserving the relevant information for the separation. It consists of a stack of convolutional layers, where each layer halves the size of the input but doubles the number of channels.

Decoder

It reconstructs and interprets the deep features and transforms it into the final spectrogram. It consists of a stack of deconvolutional layers.

Residual/skip-connections

As the encoder and decoder are symmetric i.e., feature maps at the same depth have the same shape, the U-Net adds residual/skip connections between layers of the encoder and decoder of the same depth. This refines the reconstruction by progressively providing finer-grained information from the

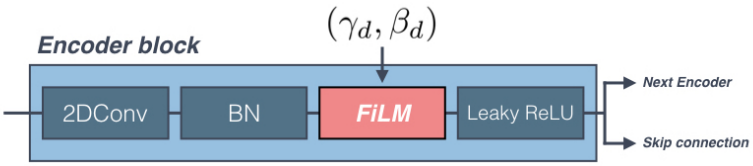


Figure 8.4: FiLM layers are placed after the batch normalisation. The output of a encoding block is connected to the next encoding block and the equivalent layer in the decoder via the residual/skip connections.

encoder to the decoder. Namely, feature maps of a layer in the encoder are concatenated to the equivalent ones in the decoder.

The final layer is **a soft mask** (sigmoid function $\in [0, 1]$) $f(X, \theta)$ which is applied to the input X to get the isolated source Y. The loss of the U-Net is defined as:

$$\mathcal{L}(X, Y; \theta) = \|f(X, \theta) \odot X - Y\|_{1,1} \tag{8.2}$$

where θ are the parameters of the system.

Architecture details

Our implementation mimics the original one (Jansson et al., 2017). The **encoder** consists in 6 encoder blocks. Each one is made of a $2D$ convolution with filters of size $(5, 5)$, stride 2, batch normalization, and leaky rectified linear units (ReLu) with leakiness 0.2. The first layer has 16 filters and we double them for each new block up to a total of 512. This defines features maps with dimensions: $(256, 64, 16)$, $(128, 32, 32)$, $(64, 16, 64)$, $(32, 8, 128)$, $(16, 4, 256)$ and $(8, 2, 512)$. The **decoder** maps the encoder, with 6 decoders blocks with stride deconvolution, stride 2 and filters of size $(5, 5)$, batch normalization, plain ReLu, and a 50% dropout in the first three blocks. The final block (the soft mask) uses a sigmoid activation. The feature maps of each decoder block have a mirror dimension regarding the **encoder** $(16, 4, 256)$, $(32, 8, 128)$, $(64, 16, 64)$, $(128, 32, 32)$, $(256, 64, 16)$, $(512, 128, 1)$. Except for the first **decoder** block, the input from the rest is the concatenation via residual/skip connections of the feature maps generated by its predecessor and the feature map of the corresponding **encoder** block. The model is trained using the ADAM optimiser (Kingma and Welling, 2014) and a 0.001 learning rate. As in (Jansson et al., 2017), we downsample to 8192 Hz, compute the Short Time Fourier Transform with a window size of 1024 and hop length of 768 frames. The input is a patch of 128 frames (roughly 11 seconds) from the normalised magnitude spectrogram for both the mixture spectrogram and the isolated instrument. The normalization is performed for the whole song not for the individual patch. We also keep the ratio between sources meaning that, if the maximum of a source before normalization is 0.86% the maximum in the mixture, it continues to be the same after the normalization.

Inserting FiLM

The U-Net has two well differentiated stages: the **encoder** and **decoder**. The **enconder** is the part that transforms the mixture magnitude input into a deep representation capturing the key elements to isolate an instrument. The **decoder** interprets this representation for reconstructing the final audio. We hypothesise that, if we can have a different way of encoding each instrument i.e., obtaining different deep representations, we can use a common 'universal' decoder to interpret all of them. Following this reasoning, we decided to condition only the U-Net encoder part. In the C-U-Net, a FiLM layer is inserted inside each encoding block after the batch normalisation and before the Leaky ReLu, as described in Figure 8.4. This decision relies on previous works where features are modified after the normalisation (Perez and Wang, 2017; de Vries et al., 2017; Kim et al., 2017). Batch normalisation normalises each feature map so that it has zero mean and unit variance (Ioffe and Szegedy, 2015). Applying FiLM after batch normalisation re-scale and re-shift feature maps after the activations. This allows the net to specialise itself to different tasks. As the output of our encoding blocks is transformed by the FiLM layer the data that flows through the residual/skip connections carries on also the transformations. If we use the *FiLM complex* layer, the control mechanism/condition generator needs to generate 2016 parameters (1008 γ_d and 1008 β_d). On the other hand, *FiLM simple* layers imply 12 parameters: one γ_d and one β_d for each of the 6 different encoding blocks, which means 2002 parameters less than for *FiLM complex* layers.

8.5.2 Condition generator: Embedding nets

The control mechanism/condition generator computes the $\gamma_d(\overline{z})$ and $\beta_d(\overline{z})$ that modify our standard U-Net behavior. Its architecture has to be flexible and robust to generate the best possible parameters. It has also to be able to find relationships between instruments. That is to say, we want it to produce similar γ_d and β_d for instruments that have similar spectrogram characteristics. Hence, we explore two different embeddings: a fully connected version and a convolutional one (CNN). Each one is adapted for the *FiLM complex* layer as well as for the *FiLM simple* layer. In every control mechanism/condition generator configuration, the last layer is always made of two concatenated fully connected layers. Each one has as many parameters (γ_d's or β_d's) as needed. With this distinction we can control γ_d and β_d individually (different activations).

Fully-connected embedding (F)

Fully-connected embedding (F) is formed of a first dense layer of 16 neurons and two fully connected blocks (dense layer, 50% dropout and batch normalization) with 64 and 256 neurons for *FiLM simple* and 256 and 1024 for *FiLM complex*. All the neurons have ReLu activations. The last fully connected block is connected with the final control mechanism/condition generator layer i.e., the two fully connected ones (γ_d and β_d). We call *C-U-Net-SiF* and *C-U-Net-CoF*, respectively, the

Table 8.1: Parameters number in millions. With separated U-Nets, each task needs a model with 10M params. C-U-Nets are multi-task and the number of params remains constant. SiF = Simple fully connected embedding, CoF = Complex fully connected embedding, SiC= Simple CNN embedding, CoC= Complex CNN embedding

MODEL	Non-conditioned	SiF	CoF	SiC	CoC
PARAM	39,30 M (4 tasks x 9,825 M)	9,85 M	12 M	9,84 M	10,42 M

C-U-Net that uses each one of these architectures.

CNN embedding (C)

Similarly to the previous one and inspired by (Shen et al., 2018), CNN embedding (C) consists in a 1D convolution with $lenght(\overline{z})$ filters followed by two convolution blocks (1D convolution with also $lenght(\overline{z})$ filters, 50% dropout and batch normalization). The first two convolutions have 'same' padding and the last one, 'valid'. Activations are also ReLu. The number of filters are 16, 32 and 64 for the *FiLM simple* version and 32, 64, 256 for the *FiLM complex* one. Again, the last CNN block is connected with the two fully connected ones. The C-U-Net that uses these architectures are called *C-U-Net-SiC* and *C-U-Net-CoC*. This embedding is specially designed for dealing with several instruments because it seems more appropriated to find common γ_d and β_d values for similar instruments.

The various control mechanisms only introduce a reduced number of parameters to the standard U-Net architecture remaining constant regardless of the instruments to separate (see Table 8.1). Additionally, they make direct use of the commonalities between instruments.

8.6 Evaluation

8.6.1 Evaluation protocol

Our objective is to prove that conditioned learning via FiLM (generic model+control) allows us to transform the U-Net into a multi-task system without losing performances with respect to each separated model. In Section 8.6.1 we review our experiment design aspects and we detail the experiment to validate the multi-task capability of the U-Net in Section 8.6.2.

Dataset

We use the Musdb18 dataset (Rafii et al., 2017). It consists of 150 tracks with a defined split of 100 tracks for training and 50 for testing. In Musdb18, mixtures are divided into four different sources: **Vocals**, **Bass**, **Drums** and **Rest** of instruments. The 'Rest' task mixes every instrument that it is not vocal, bass or drums. From the 100 tracks, we use 95 (randomly assigned) for training, and the remaining 5 for the validation set, which is used for early stopping. The performance is evaluated

on the 50 test tracks. Consequently, the U-Net is trained for four tasks (one task per instrument) and $\overline{z}$ has four elements.

Audio Reconstruction method

The system works exclusively on the magnitude of audio spectrograms. The output magnitude is obtained by applying the mask to the mixture magnitude. As in (Jansson et al., 2017), the final predicted source (the isolated audio signal) is reconstructed concatenating temporally (without overlap) the output magnitude spectra and using the original mix phase unaltered. We compute the predicted accompaniment subtracting the predicted isolated signal to the original mixture. Despite there are better phase reconstruction techniques such as (Mayer et al., 2017), errors due to this step are common to both methods (U-Net and C-U-Net) and do not affect our main goal: to validate conditioned learning for source separation.

Evaluation metrics

We evaluate the performances of the separation using the mir_evaltoolbox (Raffel et al., 2014). We compute three metrics: Source-to-Interference Ratios (SIR), Source-to-Artifact Ratios (SAR) and Source-to-Distortion Ratios (SDR) (Vincent et al., 2006). These metrics compare each estimated source $\hat{s}_j$ to its given true source s_j. They first decompose $\hat{s}_j$ into four components: $\hat{s}_j = s_{target} + e_{interf} + e_{noise} + e_{artif}$ where s_{target} is the part of estimated source that comes from the orginal target source and the interference (error coming from other unwanted sources), distortion (sensor noise, spatial or filtering distortion) and artifacts (other causes like forbidden distortions of the sources and/or 'burbling' artifacts) error terms produced by the process (Vincent et al., 2006). The metrics compute different energy ratios to evaluate the relative amount of each of these four terms. In practice, SIR measures the interference from other sources, SAR the algorithmic artifacts introduce in the process and SDR resumes the overall performance:

$$SIR := 10 \cdot log_{10} \frac{||s_{target}||^2}{||e_{interf}||^2}$$
$$SAR := 10 \cdot log_{10} \frac{||s_{target} + e_{noise} + e_{artif}||^2}{||e_{artif}||^2}$$
$$SDR := 10 \cdot log_{10} \frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2}$$

To compute the three measures we need the define the 'accompaniment' i.e. the mixture part that does not correspond to the target source. Each task has a different accompaniment e.g., for the drums the accompaniment is rest+vocals+bass. We create the accompaniments by adding the

audio signal of the needed sources. These metrics are measured in dB.

Activation function for γ_d and β_d

One of the most important design choices is the activation function for γ_d and β_d. We tested all the possible combinations of three activation functions (linear, sigmoid and tanh) in the *C-U-Net-SiF* configuration. As in (Perez and Wang, 2017), the U-Net works better when γ_d and β_d are linear. Hence, our γ_d's and β_d's have always linear activations.

Training flexibility

The conditioning mechanism gives the flexibility to have continuous values in the input $\overline{z} \in [0, 1]$, which weights the target output Y by the same value. We call this training method **progressive**. In practice, while training, we randomly weight $\overline{z}$ and Y by a value between 0 and 1 every 5 instances. This is a way of dealing with ablations by making the control mechanism robust to noise. Additionally, it is a natural way of doing data augmentation. As shown in Table 8.2, this training procedure *(p)* improves the models. Thus, we adopt it in our training. Moreover, preliminary results (not reported) show that the U-Net can be trained for complex tasks like bass+drums or voice+drums. These complex tasks could benefit from 'in between-class learning' method (Tokozume et al., 2018) where $\overline{z}$ will have different intermediate instrument combinations.

8.6.2 Multitask experiment

We want to prove that a given U-Net can isolate the **Vocals**, **Drums**, **Bass**, and **Rest** as good as four separated U-Net trained specifically for each task[2] We call this set of separated U-Net, *Fix-U-Nets*. Each U-Nets version (one model) is compared with the Fix-U-Nets set (four models). We review the results at Table 8.2 and show a comparison per task in Table 8.3.

Results in Table 8.2 for all 4 instruments highlight that *FiLM simple* layers work as good as the complex ones. This is quite interesting because it means that applying 6 affine transformations with just 12 scalars (6 γ_i and 6 β_i) at a precise point allows the C-U-Net to do several source separations. With *FiLM complex* layers it is intuitive to think that treating each feature map individually let the C-U-Net learn several deep representations in the encoder. However, we have no intuitive explanation for *FiLM simple* layers. We did the Tukey test with no significant differences between the *Fix-U-Nets* and the *C-U-Nets* for any task and metric. Another remark is that the four C-U-Nets benefit from the *progressive* training. Nevertheless, it impacts more the simple layers than in the complex ones. We think that the restriction of the former (fewer parameters) helps them to find an optimal state.

[2]with the same learning rate and optimizer as the U-Nets.

Table 8.2: Overall performance (mean ± std) for the avegare value over the 4 tasks. *Si*= simple FiLM, *Co*= complex FiLM, *F*= Fully-embedded and *C*= CNN-embedded, *p*= progressive training or *np*= non-progressive training.

MODEL	Total		
	SIR	SAR	SDR
Fix-U-Net(x4)	7.31 ± 4.04	5.70 ± 3.10	2.36 ± 3.96
C-U-Net-SiC-np	7.35 ± 4.13	5.74 ± 3.18	2.34 ± 3.69
C-U-Net-SiC-p	**8.00** ± 4.37	**5.74** ± 3.63	**2.54** ± 4.07
C-U-Net-CoC-np	7.27 ± 4.24	5.60 ± 2.88	2.36 ± 3.81
C-U-Net-CoC-p	7.49 ± 4.54	5.67 ± 3.03	2.42 ± 4.21
C-U-Net-SiF-np	7.23 ± 3.97	5.59 ± 3.01	2.22 ± 3.67
C-U-Net-SiF-p	7.64 ± 4.05	5.73 ± 2.88	2.46 ± 3.88
C-U-Net-CoF-np	7.42 ± 4.20	5.59 ± 3.07	2.32 ± 3.85
C-U-Net-CoF-p	7.52 ± 4.04	5.71 ± 2.99	2.42 ± 3.97

However, these results do not prove nor discard the significant similarity between systems. For demonstrating that, we have carried out a Pearson correlation experiment. The results are detailed in Figure 8.5. The Pearson coefficient measures the linear relationship between two sets of results (+1 implies an exact linear relationship). It also computes the p-value that indicates the probability that uncorrelated systems have produced them. Our distinct C-U-Net configurations have a global $corr > .9$ and p-value < 0.001. Which means that there is always more than 90% correlation between the performance of the four separated *U-Nets* and the (various) conditional version(s). Additionally, there is almost no probability that a C-U-Net version is not correlated with the separated models. We have also computed the Pearson coefficient and p-value per task and per metric with the same results. In Figure 8.5 shows a strong correlation between the *Fix-U-Net* results and the distinct *C-U-Nets* (independently of the task or metric). Thus, if one works well, the others too and vice versa.

In Table 8.3 we detail the results per task and metric for the *Fix-U-Net* and the *C-U-Net-CoF* which is not the best C-U-Net but the one with the highest correlation with the separated ones. There we can see how their performances are almost identical. Nevertheless, our vocal isolation (in any case) is not as good as the one reported in (Jansson et al., 2017), we believe that this is mainly due to the lack of data. These results can only be compared with the Wave-U-Net (Stoller et al., 2018c). Although they report the results (only the SDR) for the four tasks in the multi-instrument version (multiple outputs layers) they only have a separated version for vocals. For vocal separation, the performance of the multi-instrument version decreases more than 2.5 dB in mean, 1.5 dB in the median and the std increase in almost 2 dB. Furthermore, the C-U-Net performs better than the multi-instrument in three out of four tasks (vocals, bass, and drums)[3]. For the 'Rest' task, the multi-instrument Wave-U-Net outperforms our C-U-Nets. This is normal because the separated

[3]Our experiment conditions are different in training data size (95 Vs 75) and in sampling rate (8192 Hz Vs 22050 Hz) than Wave-U-Net.

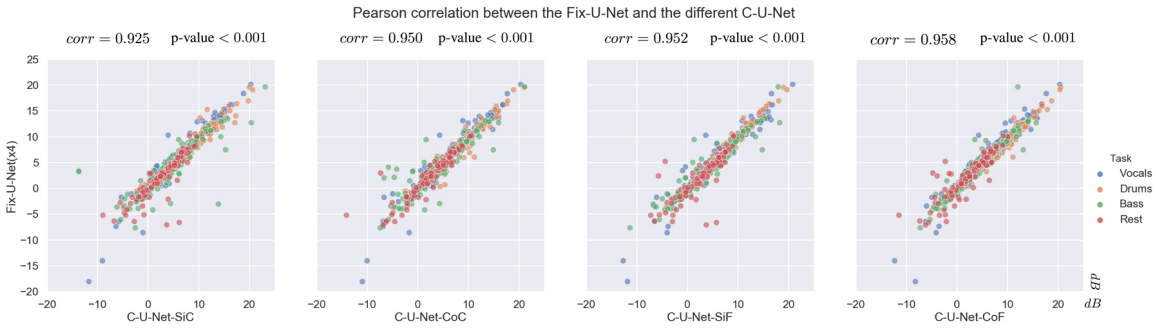


Figure 8.5: Each graph correlates the performance of two models. On top of it, we show the correlation and p-value. The 'y' axis represents the fixed version (the four separated U-Nets) and the 'x' one a different C-U-Net version (with progressive train). The coordinates of each dots correspond to the models' performance i.e., 'y' position for the Fix-U-Net performance and 'x' for C-U-Net. There are three dots per song, one per metric (SIR, SAR, and SDR) which does a total of 600 (50 songs x 3 metrics x 4 instruments). The dot alignment in the diagonal implies a strong correlation between models: if one works well, so do the others, and vice versa. Each color highlights the points of each source separation task.

Table 8.3: Task comparison between the *separated U-Nets* and the *C-U-Net-CoF*. Results indicate that they perform similarly for all the tasks. We also add the multi-instrument Wave-U-Net (M) results (median in parenthesis) and when possible the separated version (D). For vocals isolation the Wave-U-Net-M performs worse than the Wave-U-Net-D (from a mean 0.55 and median 4.58 to -2.10 and 3.0).

	Model	SIR	SAR	SDR
Vocals	*Fix-U-Net(x4)*	10.70 ± 4.26	5.39 ± 3.58	3.52 ± 4.88 (4.72)
	C-U-Net-CoF	10.76 ± 4.39	5.32 ± 3.27	3.50 ± 4.37 (4.65)
	Wave-U-Net-D	-	-	0.55 ± 13.67 (4.58)
	Wave-U-Net-M	-	-	-2.10 ± 15.41 (3.0)
Drums	*Fix-U-Net(x4)*	10.08 ± 4.28	6.42 ± 3.28	4.28 ± 3.65 (4.13)
	C-U-Net-CoF	10.03 ± 4.34	6.80 ± 3.25	4.30 ± 3.81 (4.38)
	Wave-U-Net-M	-	-	2.88 ± 7.68 (4.15)
Bass	*Fix-U-Net(x4)*	4.64 ± 4.76	6.51 ± 2.68	1.46 ± 4.31 (2.48)
	C-U-Net-CoF	5.30 ± 4.73	6.29 ± 2.39	1.65 ± 4.07 (2.60)
	Wave-U-Net-M	-	-	-0.30 ± 13.50 (2.91)
Rest	*Fix-U-Net(x4)*	3.83 ± 2.84	4.47 ± 2.85	0.19 ± 3.00 (0.97)
	C-U-Net-CoF	4.00 ± 2.70	4.37 ± 3.06	0.24 ± 3.64 (1.71)
	Wave-U-Net-M	-	-	1.68 ± 6.14 (2.03)

U-Net has already problems with this class and the C-U-Nets inherits the same issues. We believe that they come from the vague definition of this class with many different instruments combinations at once.

This proves that the various C-U-Nets behave in the same way as the separated U-Nets for each task and metric. It also demonstrates that conditioned learning via FiLM is robust to diverse control

mechanisms/condition generators and FiLM layers. Moreover, it does not introduce any limitations which are due to other factors.

8.7 Conclusions and future work

In this chapter, we have applied conditioned learning to the problem of instrument source separations by adding a control mechanism to the U-Net architecture. The C-U-Nets can do several source separation tasks without losing performance as it does not introduce any limitation and makes use of the commonalities of the distinct instruments. It has a fixed number of parameters (much lower than the separated model approach) independently of the number of instruments to separate. We believe that conditioned learning via FiLM will benefit many MIR problems because it defines a transparent and direct way of inserting external data to modify the behavior of a network. Our key contributions are:

1. the C-U-Net, a joint model that changes its behavior depending on external data and performs for any task as good as a separated model trained for it. C-U-Net has a fixed number of parameters no matter the number of output sources.

2. The C-U-Net proves that conditioned learning via FiLM layers is an efficient way of inserting external information to control deep neural network architectures.

3. The *FiLM simple*, a new conditioning layer that works as good as the original one but requires less γ_d's and β_d's.

Conditioned learning faces problems providing a generic model and a control mechanism. This gives flexibility to the systems but introduces new challenges. We plan to extend the C-U-Net to more instruments to find its limitations and to explore the performance for complex tasks i.e., separating several instruments combinations (e.g., vocals+drums). We also intend to integrate it in other architectures such as Wave-U-Net and data augmentation techniques (Cohen-Hadria et al., 2019). Currently, the multitrack version of DALI divides the mixture into 'vocals' + 'accompaniment'. However, the rest of the sources are also presented but they not prepared yet to use (see Chapter 4.7). We plan to add these sources to work with more instruments. Likewise, we are exploring ways of adding new conditions (namely new instrument isolation) to a trained C-U-Net and how to separate the joint training, with the goal of creating a generic model than can be easily adapted to several control mechanisms.

Chapter 9

Vocal Source Separation

Informed source separation has recently gained renewed interest with the introduction of neural networks and the availability of large multitrack datasets containing both the mixture and the separated sources. These approaches use prior information about the target source to improve separation. Historically, MIR researchers have focused primarily on score-informed source separation, but more recent approaches explore lyrics-informed source separation. However, because of the lack of multitrack datasets with time-aligned lyrics, models use weak conditioning with the non-aligned lyrics. In this chapter, we present a multimodal multitrack dataset with lyrics aligned in time at the phoneme level as well as explore strong conditioning using the aligned phonemes. Our model explores the C-U-Net architecture and takes as input both the magnitude spectrogram of a musical mixture and a matrix with aligned phoneme information. The phoneme matrix is embedded to obtain the parameters that control FiLM layers. These layers condition the C-U-Net feature maps to adapt the separation process to the presence of different phones via affine transformations We show that phoneme conditioning can be successfully applied to improve singing voice source separation.

9.1 Introduction

Music source separation aims to isolate the different instruments that appear in an audio mixture (a mixed music track), reversing the mixing process. Informed-source separation uses prior information about the target source to improve separation. Researchers have shown that deep neural architectures can be effectively adapted to this paradigm (Kinoshita et al., 2015; Miron et al., 2017). Music source separation is a particularly challenging task. Instruments are usually correlated in time and frequency with many different harmonic instruments overlapping at several possitions and with dynamics variations. Without additional knowledge about the sources the separation is often infeasible. To address this issue, MIR researchers have integrated into the source separation process prior knowledge about the different instruments presented in a mixture, or musical scores that

indicate where sounds appear. This prior knowledge improves the performances (Slizovskaia et al., 2020; Ewert et al., 2014; Miron et al., 2017). Recently, conditioning learning has shown that neural networks architectures can be effectively controlled for performing different music source isolation tasks (Meseguer-Brocal and Peeters, 2019; Tzinis et al., 2019; Slizovskaia et al., 2019; Seetharaman et al., 2019; Samuel et al., 2020; Schulze-Forster et al., 2019)

Various multimodal context information can be used. Although MIR researchers have historically focused on score-informed source separation to guide the separation process, lyrics-informed source separation has become an increasing research area (Chandna et al., 2020; Schulze-Forster et al., 2019). Singing voice is one of the most important elements in a musical piece (Demetriou et al., 2018). Singing voice tasks (e.g. lyric or note transcription) are particularly challenging given its variety of timbre and expressive versatility. Fortunately, recent data-driven machine learning techniques have boosted the quality and inspired many recent discovers (Gómez et al., 2018; Humphrey et al., 2018). Singing voice works as a musical instrument and at the same time conveys a semantic meaning through the use of language (Humphrey et al., 2018). The relationship between sound and meaning is defined by a finite phonetic and semantic representations (Goldsmith, 1976; Ladd, 2008). Singing in popular music usually has a specific sound based on phonemes, which distinguishes it from the other musical instruments. This motivates researchers to use prior knowledge such as a text transcript of the utterance or linguistic features to improve the singing voice source separation (Chandna et al., 2020; Schulze-Forster et al., 2019). However, the lack of multitrack datasets with time-aligned lyrics has limited them to develop their ideas and only weak conditioning scenarios have been studied i.e. using the context information without explicitly informing where it occurs in the signal. Time-aligned lyrics provide abstract and high-level information about the phonetic characteristics of the singing signal. This prior knowledge can facilitate the separation and be beneficial to the final isolation.

Looking for combining the power of data-driven models with the adaptability of informed approaches, we propose a multitrack dataset with time-aligned lyrics. Then, we explore how we can use strong conditioning where the content information about the lyrics is available frame-wise to improve vocal sources separation (see part 4.7). We investigate strong and weak conditioning using the aligned phonemes via FiLM layer in U-Net based architecture (see Chatper 8). We show that phoneme conditioning can be successfully applied to improve standard singing voice source separation and that the simplest strong conditioning outperforms any other scenario.

9.2 Formalization

We use the multimodal information as context to guide and improve the separation. We formalize our problem satisfying certain properties summarized as (Bengio et al., 2013) (see part 1.1):

How is the multimodal model constructed? We divide the model into two distinct parts (Dumoulin et al., 2018): a *generic* network that carries on the main computation and a *control mechanism* that conditions the computation regarding context information and adds additional flexibility. The *conditioning* itself is performed using Feature-wise Linear Modulation (FiLM) layers (Perez and Wang, 2017). FiLM can effectively modulate a generic source separation model by some external information, controlling a single model to perform different instrument source separations (Meseguer-Brocal and Peeters, 2019; Slizovskaia et al., 2020). With this strategy, we can explore the *control* and *conditioning* parts regardless of the *generic* network used.

When is the context information used? We can see this as at which depth in the *generic* network we insert the context information, and when it affects the computation i.e. weak (or strong) conditioning without (or with) explicitly informing where it occurs in the signal.

What context information? We explore here prior information about the phonetic evolution of the singing voice, aligned in time with the audio. To this end, we introduce a novel multitrack dataset with lyrics aligned in time.

9.3 Related work

Informed source separation uses context information about the sources to improve the separation quality, introducing in models additional flexibility to adapt to observed signals. Researchers have explored different approaches for integrating different prior knowledge in the separation (Liutkus et al., 2013). In this section we review previous works related to informed source separation in speech and then in music, where we review both score-informed and text-informed.

9.3.1 In speech

LeMagoarou et al. (2015) present one of the first text-informed source separation approaches. They propose a speech example-based paradigm where the text information generates (via synthesizer or human) a speech example aligned with the original mixture using DTW. The example guides the separation exploiting linguistic similarities between the target speech and the example speech signal. Kinoshita et al. (2015) use text features derived from forced-aligned phonemes with noisy speech together with the audio features to train a deep neural network that predicts enhanced spectrum parameters. The authors show that distortion in the separation is smaller when using text. Automatic Speech Recognition (ASR) can directly identify phonemes at each frame from the mixture without using text-transcript (Chazan et al., 2016; Wang et al., 2016). Then, pre-trained phoneme-specific networks perform the separation. Biadsy et al. (2019) uses an end-to-end sequence-to-sequence encoder/decoder architecture with an additional ASR decoder to predict the (grapheme or phoneme) transcript, which conditions the encoder latent representation. Although its primary

application is voice conversion, it is useful in a source extraction scenario. However, it requires a dataset of parallel paired input-output speech utterances. Schulze-Forster et al. (2020) describe a multitask model that jointly perform text-informed speech separation and phoneme alignment. Their model uses a bidirectional recurrent neural network (BRNN) where the context information is extracted from the text via an attention mechanism. The context information is refined with an additional loss for solving phoneme alignment task. They show that jointly solving both tasks leads to mutual benefits. Takahashi et al. (2020) explicitly incorporates the phonetics using transfer learning. They first train an ASR encoder/decoder model on a large clean speech corpus. They adapt the intermediate features obtained from the encoder into a suitable representation for voice separation using a domain translation network. These features condition the separation of a net based on Tasnet (Luo and Mesgarani, 2019).

9.3.2 In music

Most of the recent data-driven music source separation methods use weak conditioning with prior knowledge about the different instruments presented in a mixture (Slizovskaia et al., 2020; Meseguer-Brocal and Peeters, 2019; Slizovskaia et al., 2019; Seetharaman et al., 2019; Samuel et al., 2020). Strong conditioning has been primarily used in score-informed source separation. In this section, we review works related to this topic as well as novel approaches that explore lyrics-informed source separation.

9.3.3 Score-informed music source separation

Scores provide prior knowledge for source separation in various ways. For each instrument (source), it defines which notes are played at which time, which can be linked to audio frames. This information can be used to guide the estimation of the harmonics of the sound source at each frame (Ewert et al., 2014; Miron et al., 2017). Pioneer approaches rely on non-negative matrix factorization (NMF). Note activities in the score can constrain the NMF-based model by setting to zero the harmonic values outside the frequency range of each nominal musical note (Ewert and Müller, 2012). Authors introduce a multi-excitation per instrument (MEI) source-filter NMF model that uses pre-learned timbre models for each instrument (Duan and Pardo, 2011; Rodriguez-Serrano et al., 2015). They also learn the NMF components used for the isolation on synthetic signals with temporal and harmonic constraints generated from the score (Fritsch and Plumbley, 2013). These methods assume that the audio is synchronized with the score and use different alignment techniques to achieve this. Nevertheless, alignment methods introduce errors. Local misalignments influence the quality of the separation (Duan and Pardo, 2011; Miron et al., 2015). This is compensated by allowing a tolerance window around note onsets and offsets (Ewert and Müller, 2012; Fritsch and Plumbley, 2013) or with context-specific methods to refine the alignment (Miron et al., 2016). Current approaches use deep neural network architectures and use the scores to filter the spectrograms, generating masks

for each source (Miron et al., 2017). The score-filtered spectrum is used as input to an encoder-decoder Convolutional Neural Network (CNN) architecture similar to (Chandna et al., 2017). Ewert and Sandler (2017) propose an unsupervised method where scores guide the representation learning to induce structure in the separation that adds class activity penalties and structured dropout extensions to the encoder-decoder architecture. Class activity penalties capture the uncertainty about the target label value and structured dropout uses labels to enforce a specific structure, canceling activity related to unwanted note.

9.3.4 Text-informed music source separation

Due to the importance of singing voice in a musical piece (Demetriou et al., 2018), it is one of the most useful source to separate in a music track. Researchers have integrated the vocal activity information to constrain a robust principal component analysis (RPCA) method, applying a vocal/non-vocal mask or ideal time-frequency binary mask (Chan et al., 2015). Schulze-Forster et al. (2019) propose a bidirectional recurrent neural networks (BRNN) method that includes context information extracted from the text via attention mechanism. The method takes as input a whole audio track and its associated text information and learn alignment between mixture and context information that enhance the separation. Recently, (Chandna et al., 2020) extract a representation of the linguistic content related to cognitively relevant features such as phonemes (but they do not explicitly predict the phonemes) in the mixture. The linguistic content guides the synthesis of the vocals.

9.4 Methodology

Our method adapts the C-U-Net architecture (Meseguer-Brocal and Peeters, 2019) to the singing voice separation task, exploring how to use the prior knowledge defined by the phonemes to improve the vocal separation. Let $X \in \mathbb{R}^{T \times M}$ be the magnitude of the Short-Time Fourier Transform (STFT) with $M = 512$ frequency bands and T time frames. We compute the STFT on an audio signal down-sampled at 8192 Hz using a window size of 1024 samples and a hop size of 768 samples. Let $Z \in \mathbb{R}^{T \times P}$ be the aligned phoneme activation matrix with $P = 40$ phoneme types as defined in the Carnegie MellonPronouncing Dictionary (CMUdict)[1] and T the same time frames as in X. For computing this matrix we use the phoneme information $A_{phoneme} = (a_{k,phoneme})_{k=1}^{K_{phoneme}}$ (see Chapter 4) for the multitrack version of DALI (see Chapter 4.7). After selecting the desired time resolution, we can derive a time frame based phoneme context activation matrix Z, which is a binary matrix that indicates the phoneme activation over time. Note that the corresponding phonemes are active during all segments defined by $(t_{min}, t_{max})_k$ We add an extra row with the 'non-phoneme' activation with 1 at time frames with no phoneme activation and 0 otherwise. Figure 9.1 illustrates the final activation matrix.

[1] urlhttps://github.com/cmusphinx/cmudict

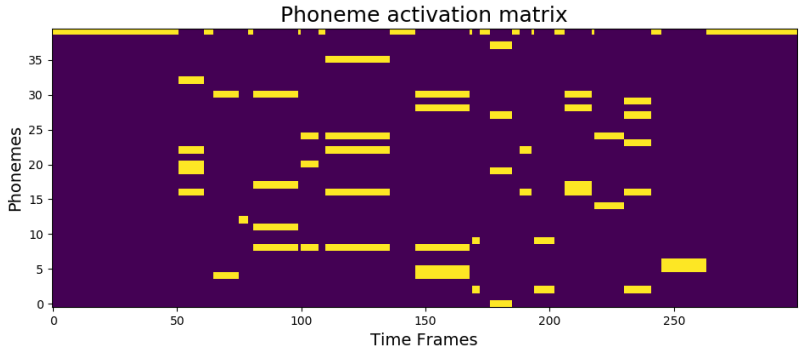


Figure 9.1: Binary phoneme activation matrix. Note how words are represented as a bag of simultaneous phonemes.

Our model consider the music track as a sucession of patch segments of duration N. It takes as inputs two submatrix $x \in \mathbb{R}^{N \times M}$ and $z \in \mathbb{R}^{N \times P}$ of $N = 128$ frames (11 seconds) derived from X and Z. The C-U-Net model has two components: a **conditioned network** that processes x and a **control mechanism** that conditions the computation with respect to z. We denote by $x_d \in \mathbb{R}^{W \times H \times C}$ the intermediate features of the **conditioned network**, at a particular depth d in the architecture. W and H represent the 'time' and 'frequency' dimension and C the number of feature channels (or feature maps). A *FiLM* layer conditions the network computation by applying an affine transformation to x_d:

$$FiLM(x_d) = \gamma_d(z) \cdot x_d + \beta_d(z) \tag{9.1}$$

where $\odot$ denotes the element-wise multiplication and $\gamma_d(z)$ and $\beta_d(z)$ are learnable parameters with respect to the input context z. A *FiLM* layer can be inserted at any depth of the original model and its output has the same dimension as the x_d input, i.e. $\in \mathbb{R}^{W \times H \times C}$. To perform this, $\gamma_d(z)$ and $\beta_d(z)$ must have the same dimensionaly as x_d, i.e. $\in \mathbb{R}^{W \times H \times C}$. However, we can define them omitting some dimensions. This results in a non-matching dimensionality with x_d, solved by broadcasting (repeating) the existing information to the missing dimensions.

As in (Jansson et al., 2017; Stoller et al., 2018c; Meseguer-Brocal and Peeters, 2019), we use a standard U-Net as **conditioned network**. This model follows an encoder-decoder mirror architecture based on CNN blocks with skip connections between layers at the same hierarchical level in the encoder and decoder. Each convolutional block in the encoder halves the size of the input and doubles the number of channels. The decoder is made of a stack of transposed convolutional operation, its output has the same size as the input of the encoder. Following the original C-U-Net architecture, we insert the *FiLM* layers at each encoding block after the batch normalization and before the Leaky ReLU (Meseguer-Brocal and Peeters, 2019).

Model	U-Net	W_{si}	W_{co}	S_{fv}	S_{fv*}	S_{cs}	S_{cs*}	S_{fs}	S_{fs*}	S_{rs}	S_{rs*}
θ	$9.83 \cdot 10^6$	$+14,060$	$+2.35 \cdot 10^6$	$+1.97 \cdot 10^6$	$+327,680$	$+80,640$	$+40,960$	$+40,320$	$+640$	$+480$	$+80$

Table 9.1: Number of parameters (θ) for the different configurations. We indicate the increase in the number of parameters w.r.t. the baseline U-Net architecture.

9.4.1 Control mechanism for weak conditioning

Weak conditioning refers to the cases where

- $\gamma_d(z)$ and $\beta_d(z) \in \mathbb{R}^1$: they are scalar parameters applied independently of the times W, the frequencies H and the channel C dimensions. They depend only on the depth d of the layer within the network (Meseguer-Brocal and Peeters, 2019).
- $\gamma_d(z)$ and $\beta_d(z) \in \mathbb{R}^C$: this is the orginal configuration proposed by (Perez and Wang, 2017) with different parameters for each channel $c \in 1, ..., C$.

We call them *FiLM simple* (W_{si}) and *FiLM complex* (W_{co}) respectively. Note how they apply the same transformation without explicitly informing where it occurs in the signal (same value over the dimension W and H) (Meseguer-Brocal and Peeters, 2020).

Starting from the context matrix $z \in \mathbb{R}^{N \times P}$, we first apply the autopool layer proposed by (McFee et al., 2018)[2] to reduce the input matrix to a time-less vector. We then fed this vector into a dense layer and two dense blocks each composed by a dense layer, 50% dropout and batch normalization. For *FiLM simple*, the number of units of the dense layers are 32, 64 and 128. For *FiLM simple*, they are 64, 256 and 1024. All neurons have ReLU activations. The output of the last block is then used to feed two parallel and independent dense layers with linear activation which outputs all the needed $\gamma_d(z)$ and $\beta_d(z)$. While for the *FiLM simple* configuration we only need 12 γ_d and β_d (one γ_d and β_d for each of the 6 different encoding blocks) for the *FiLM complex* we need 2016 (the encoding blocks feature channel dimensions are 16, 32, 64, 128, 256 and 512, which adds up to 1008).

9.4.2 Control mechanism for strong conditioning

In this section, we extend the original *FiLM* layer mechanism to adapt it to the strong conditioning scenario. The context information represented in the input matrix z describes the presence of the phonemes $p \in \{1, \ldots, P\}$ over time $n \in \{1, \ldots N\}$. As in the popular Non-Negative Matrix factorization (Lee and Seung, 2001) (but without the non-negativity constraint), our idea is to represent this information as the product of tensors: an activation and two basis tensors (Meseguer-Brocal and Peeters, 2020).

The **activation tensor** z_d indicates which phoneme occurs at which time: $z_d \in \mathbb{R}^{W \times P}$ where W is the dimension which represents the time at the current layer d (we therefore need to map the

[2]The auto-pool layer is a tuned soft-max pooling that automatically adapts the pooling behavior to interpolate between mean and max-pooling for each dimension

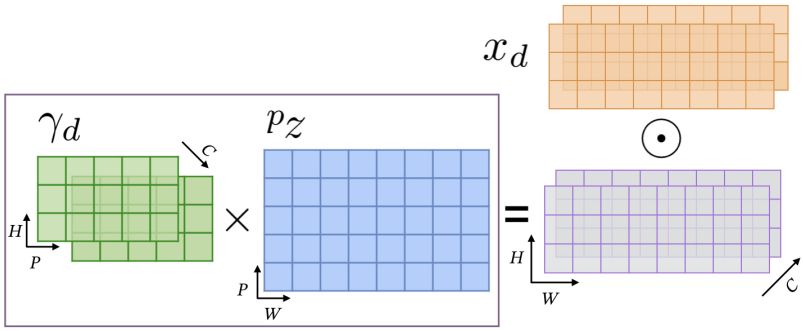


Figure 9.2: Strong conditioning example with $(\gamma_d \times z_d) \odot x_d$. The phoneme activation z_d defines how the basis tensors (γ_d) are employed for performing the conditioning on x_d.

time range of z to the one of the layer d) and P the number of phonemes.

The **two basis tensors γ_d and $\beta_d \in \mathbb{R}^{H \times C \times P}$** where H is the dimension which represents the frequencies at the current layer d, C the number of input channels and P the number of phonemes. In other words, each phoneme p is represented by a matrix in $\mathbb{R}^{H \times C}$. This matrix represents the specific conditioning to apply to x_d if the phoneme exists (see Figure 9.2). These matrices are learnable parameters (neurons with linear activations) but they do not depend on any particular input information (at a depth d they do not depend on x nor z), they are rather "activated" by z_d at specific times. As for the 'weak' conditionning, we can define different versions of the tensors

- the **full-version** (S_{fv}) (described so far) which has three dimensions: $\gamma_d, \beta_d \in \mathbb{R}^{H \times C \times P}$
- the **channel simple version** (S_{cs}): each phoneme is represented by a vector over input channels (therefore constant over frequencies): $\gamma_d, \beta_d \in \mathbb{R}^{C \times P}$
- the **frequency simple version** (S_{fs}): each phoneme is represented by a vector over input frequencies (therefore constant over channels): $\gamma_d, \beta_d \in \mathbb{R}^{H \times P}$
- the **really-simple version** (S_{rs}): each phoneme is represented as a scalar (therefore constant over frequencies and channels): $\gamma_d, \beta_d \in \mathbb{R}^{P}$

The global conditioning mechanism can then be written as

$$FiLM(x_d, z_d) = (\gamma_d \times z_d) \odot x_d + (\beta_d \times z_d) \tag{9.2}$$

where $\odot$ is the element-wise multiplication and $\times$ the matrix multiplication. We broadcast γ_d and β_d for missing dimensions and transpose them properly to perform the matrix multiplication. We test two different configurations: inserting FiLM at each encoder block as suggested in (Meseguer-Brocal and Peeters, 2019) and inserting FiLM only at the last encoder block as proposed at (Slizovskaia et al., 2020). We call the former 'complete' and the latter 'bottleneck' and denote it with $*$ after the model acronym. We resume the different configurations at Table 9.1.

	Train	Val	Test
Threshold	$.88 > NCC >= .7$	$.89 > NCC >= .88$	$.89 > NCC$
Songs	357	30	101

Table 9.2: DALI split according to agreement score NCC.

9.5 Experiments

9.5.1 Training

DATA. As described in Table 9.2, we split the multitrack version of DALI into three sets according to the normalized agreement score NCC (see chapter 4.3). This score provides a global indication of the global alignment correlation between the annotations and the vocal activity.

DETAILS. We train the model using batches of 128 spectrograms randomly drawn from the training set with 1024 batches per epoch. The loss function is the mean absolute error between the predicted vocals (masked input mixture) and the original vocals. We use a learning rate of 0.001 and the reduction on plateau and early stopping callbacks evaluated on the validation set. We set the 'patience' parameter to 15 and 30 respectively and a min delta variation for early stopping to $1e-5$. Our output is a Time/Frequency mask to be applied to the magnitude of the input STFT mixture. We use the phase of the input STFT mixture to reconstruct the waveform with the inverse STFT algorithm.

For the strong conditioning, we apply a `softmax` on the input phoneme matrix z over the phoneme dimension P to constrain the outputs to sum to 1, meaning it lies on a hyperplane, which helps in the optimization.

9.5.2 Evaluation metrics

We evaluate the performances of the separation using the mir_evaltoolbox (Raffel et al., 2014). As in chapter 8.6.1, we compute three metrics: Source-to-Interference Ratios (SIR), Source-to-Artifact Ratios (SAR), and Source-to-Distortion Ratios (SDR) (Vincent et al., 2006). In practice, SIR measures the interference from other sources, SAR the algorithmic artifacts introduce in the process and SDR resumes the overall performance. We obtain them globally for the whole track. However, these metrics are ill-defined for silent sources and targets. Hence, we compute also the Predicted Energy at Silence (PES) and Energy at Predicted Silence (EPS) scores (Schulze-Forster et al., 2019). PES is the mean of the energy in the predictions at those frames with silent target and EPS is the opposite, the mean of the target energy of all frames with silent prediction and non-silent target. For numerical stability, in our implementation, we add a small constant $\epsilon = 10^{-9}$ which results in a lower boundary of the metrics to be -80 dB (Slizovskaia et al., 2020). We consider as silent segments those that have a total sum of less than -25 dB of the maximum absolute in the audio.

Training	Test	Aug	SDR	SIR	SAR
Musdb18 (90)	Musdb18 (50)	False	4.27	13.17	5.17
		True	4.46	12.62	5.29
DALI (357)	Musdb18 (50)	False	4.60	14.03	5.39
		True	4.96	13.50	5.92
	DALI (101)	False	3.98	12.05	4.91
		True	4.05	11.40	5.32

Table 9.3: Data augmentation experiment for the U-Net architecture.

We report in Table 9.3 the **median** values of these metrics over the all tracks in the DALI test set. For SIR, SAR, and SDR larger values indicate better performance, for PES and EPS smaller values, mean better performance.

9.5.3 Data augmentation

To augment the data, we randomly created 'fake' input mixtures every 4 real mixtures. In normal training, we employ the mixture as input and the vocals as a target. However, we do not make use of the accompaniment (which is only employed during evaluation). We can integrate it creating 'fake' inputs by automatically mixing (mixing meaning simply adding) the target vocals to a random sample accompaniment from our training set.

We test this data augmentation process using the standard U-Net architecture and checked that it improves the performance (see Table 9.3). We train two models on DALI and Musdb18 dataset (Rafii et al., 2017)[3]. This data augmentation enables models to achieve better SDR and SAR but lower SIR.

This technique does not reflect a large improvement when the model trained on DALI is tested on DALI. However, when this model is tested on Musdb18, it shows a better generalization (we have not seen any song of Musidb18 during training) than the model without data augmentation (we gain 0.36 dB). One possible explanation for not having a large improvement on DALI is the larger size of the test set. It also can be due to the fact that vocal targets in DALI still contain leaks such as low volume music accompaniment that come from the singer headphones. We adopt this technique for training all the following models.

Finally, we confirmed a common belief that training with a large dataset and clean separated sources improves the separation over a small dataset (Prétet et al., 2019). Both models trained on DALI (with and without augmentation) improve the results obtained with the models trained on Musdb18.

[3]We use 10 songs of the training set for the early stopping and reduction on plateau callbacks

Model	SDR	SIR	SAR	PES	EPS
U-Net	4.05	11.40	5.32	-42.44	-64.84
W_{si}	**4.24**	**11.78**	5.38	**-49.44**	-65.47
W_{co}	**4.24**	**12.72**	5.15	**-59.53**	-63.46
S_{fv}	4.04	**12.14**	5.13	**-59.68**	-61.73
S_{fv*}	**4.27**	**12.42**	5.26	**-54.16**	-64.56
S_{cs}	**4.36**	**12.47**	5.34	**-57.11**	-65.48
S_{cs*}	**4.32**	**12.86**	5.15	**-54.27**	**-66.35**
S_{fs}	4.10	11.40	5.24	47.75	-62.76
S_{fs*}	4.21	**13.13**	5.05	**-48.75**	**-72.40**
S_{rs}	**4.45**	**11.87**	**5.52**	**-51.76**	-63.44
S_{rs*}	**4.26**	**12.80**	5.25	**-57.37**	-65.62

Table 9.4: Median performance in dB of the different models on the DALI test set. In bold are the results that significantly improve over the U-Net ($p < 0.001$) and inside the circles the best results for each metric.

9.6 Results

We report the median source separation metrics (SDR, SAR, SIR, PES, ESP) in Table 9.4. To measure the significance of the improvement differences between the results, we performed a paired t-test between each conditioning model and the standard U-Net architecture, the baseline. This test measures (p-value) if the differences could have happened by chance. A low p-value indicates that data did not occur by chance. As expected, the improvement is consistent over most of the proposed methods. The statistical significance ($p < 0.001$) for the SDR, SIR, and PES is generalized except for the versions where the basis tensors have a 'frequency' H dimension. This is an expected result since when singing, the same phoneme can be sung at different frequencies (appearing at many frequency positions in the feature maps). Hence, these systems have difficulties to find generic basis tensors. This also explains why the 'bottleneck' versions (for both S_{fs*} and S_{fv*}) outperforms the 'complete' while this is not the case for the other versions. Most versions also improve the performance on silent vocal frames with a much lower PES. However, there is no difference in predicting silence at the right time (same EPS). The only metric that does not consistently improve is SAR, which measures the algorithmic artifacts introduced in the process. Our conditioning mechanisms can not reduce the artifacts that seem more dependent on the quality of the training examples (it is the metric which has the highest improvement in the data augmentation experiment Table 9.3). Figure 9.3 shows a comparison with the distribution of SDR, SIR, and SAR for the best model S_{rs} and the U-Net. We can see how the distributions move toward higher values.

One relevant remark is the fact that we can effectively control the network with just a few parameters. S_{rs} just adds 480 (or just 80! for S_{rs*}) new learnable parameters and have significantly better performance than S_{fv} that adds $1.97 \cdot 10^6$ parameters. We believe that the more complex control mechanisms tend to find complex basis tensors that do not generalize well. In our case, it

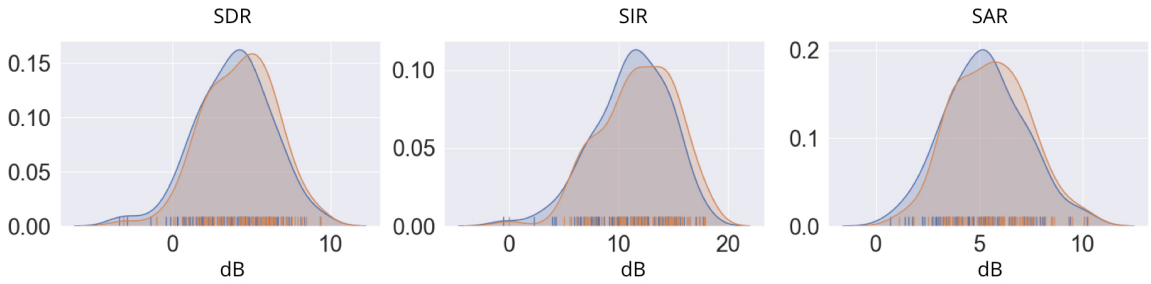


Figure 9.3: Distribution of scores for the the standard U-Net (Blue) and S_{rs} (Orange).

is more effective to perform a simple global transformation. In the case of weak conditioning, both models behave similarly although W_{si} has $1.955 \cdot 10^6$ fewer parameters than W_{co}. This seems to indicate that controlling channels is not particularly relevant.

Regarding the different types of conditioning, when repeating the paired t-test between weak and strong models only S_{rs} outperforms the weak systems. We believe that strong conditioning can lead to higher improvements but several issues need to be addressed. First of all, there are missalignments in the annotations that force the system to perform an unnecessary operation which damages the computation. This is one of the possible explanations of why models with fewer parameters perform better. They are forced to find more generic conditions. The weak conditioning models are robust to these problems since they process z and compute an optimal modification for a whole input patch (11 seconds). We also need to disambiguate the phonemes inside words. Currently, a word is described as a bag of phonemes that occur at the same time. This prevents strong conditioning models to learn properly the phonemes in isolation, instead, they consider them jointly with the other phonemes.

9.7 Conclusions

The goal of this chapter was to improve singing voice separation using the prior knowledge defined by the phonetic characteristics. We use the phoneme activation as side information and show that it helps in the separation.

In future works, we intend to use other prior aligned knowledge such as vocal notes or characters also defined in DALI. Regarding the conditioning approach and since it is transparent to the conditioned network, we are determined to explore recent state-of-the-art source separation methods such as Conv-Tasnet(Luo and Mesgarani, 2019). The current formalization of the two basis tensors γ_d and β_d does not depend on any external factor. A way to exploit the complex control mechanisms is to make these basis tensors dependent on the input mixture x which may add additional flexibility. Finally, we plan to jointly learn how to infer the alignment and perform the separation (Schulze-Forster et al., 2020; Takahashi et al., 2020).

The general idea of lyrics-informed source separation leaves room for many possible extensions. The formalization we presented relies on time-aligned lyrics which is not the real-world scenario. Features similar to the phoneme activation (Vaglio et al., 2020; Stoller et al., 2019) can replace them or be used to align the lyrics as a pre-processing step. These two options would allow adapting the current system to the real-world scenario. These feature can also help in properly placing and desambiaguating the phonemes of a word to improve the current annotations.

Chapter 10

Conclusions and Future Work

10.1 Summary of Contributions

Multimodal learning is a growing and exciting field in the MIR community that aims to understand music through its various facets. Since the earlier works, researchers have investigated and developed multimodal learning systems. Almost every MIR task can be formalized in a multimodal set up, showing that integrating the natural multidimensionality of music can improve the results (Essid and Richard, 2012; Simonetta et al., 2019). Nevertheless, it still grows at a much slower rate than other topics in the community. We hypothesize that the lack of development stems from (1) the lack of good quality multimodal datasets and (2) the costly process of integrating data from different domains (see chapter 1).

Apart from the obvious audio signal, music can be expressed in many different dimensions such as scores, videos, or motion. Among them, text is one of the most used information. It provides many different and complementary sources of information. From editorial reviews that describe music, social web content that provides user interactions to metadata-tags that group music in categories, or ratings that classify it from a subjective point of view, whether someone likes it or not. All these sources add many useful insights not only about music but also about how we interact to it. In this thesis, we focused on lyrics because they have a direct connection to the audio signal via the singing voice. The singing voice acts as a musical instrument and at the same time conveys semantic meaning through the lyrics, adding a linguistic dimension that complements the abstraction of the musical instruments. Additionally, singing voice tasks (e.g. lyrics or note transcription) are particularly challenging given its variety of timbre and expressive versatility. Getting access to complementary information helps in developing better systems.

During the development of our work we have investigated different multimodal strategies and problem formalizations. To summarize them, we outlined the four questions that guide this dissertation in Chapter1. In this section, we will outline our answers to each of those questions.

How can we obtain large amounts of labeled data where lyrics and its melodic representation aligned in time with the audio to train data-driven methods?

In Chapter 3, we highlighted the need for more multimodal datasets with lyrics and notes aligned in time, reviewing the available datasets. We saw that prior datasets were very small, inconsistent regarding the levels of alignment and without note annotations. To address this, we released the DALI dataset that contains lyrics and vocal notes aligned in time at different levels of granularity and includes a multitrack subset with separated vocal and accompaniment separated. DALI remains the largest, most complete dataset with lyrics and vocal notes aligned in time annotations to date. We outlined the different versions and the characteristics of the data, formally defined the dataset and introduced the developed tools that help us to work with this complex data.

In Chapter 4, we detailed our methodology to create DALI automatically. We leverage data from the Internet and integrate active learning and weakly-supervised learning techniques in the process. Non-expert karaoke time-aligned lyrics and notes describe the lyrics as a sequence of time-aligned notes with their associated textual information. We then linked each annotation to the correct audio and globally aligned the annotations to it using the normalized cross-correlation on the voice annotation sequence and the singing voice probability vector. We used the teacher-student paradigm to create an interaction between the dataset creation and model learning that benefits each other. This helped us to improve our SVD system which allows a better selection and global alignment.

In Chapters 7 and 9 we used this data to train and evaluate a variety of data-driven methods for lyrics segmentation and source separation. The resulting approaches sparked several new research directions.

In summary, we can obtain large amounts of aligned lyrics and notes by collecting data from the Internet and developing methods that can filter and adapt the annotations to the audio. However, this process comes with a price and it is error prone.

How can we automatically identify and solve errors in these labels?

In Chapter 5, we deepened into the different types of labeling errors presented in DALI. We proposed automatic solutions to global alignment issues with simple correlations. These are effective but limited. We explored alignment techniques to solve different local alignment issues. We treated the annotations as 'scores' employing the *musical note events* labels and aligned them with the audio. However, there is not an efficient way of measuring if the new labels are better or worse than the original ones and we cannot properly measure the different proposed methods.

To address this, we proposed in Chapter 6 a novel data cleansing technique that considers the time-varying structure of labels. We train our model in a self-supervised way to automatically tell if a note label is correct for a given audio signal. Our method exploits the local structure of the labels to find possible errors in vocal note event annotations. We evaluated this at the frame level and obtained the error probability function vector that measures the 'quality' of the annotations.

We showed that this vector can be successfully applied to improve the performance of estimating the f_0, a downstream inference task. We improved the Raw Pitch Accuracy by over 10 percentage points simply by filtering the training dataset using our data cleansing model. Our approach is particularly useful when training on very noisy datasets such as those collected from the Internet and automatically aligned. We also used our proposed error detection model to estimate the error rate, knowing the current status of the dataset.

How can we exploit the inherent relationships between lyrics and audio to improve the performance for lyrics segmentation?

In Chapter 7, we used DALI for improving lyrics segmentation using a model-agnostic and early fusion approach that integrates the audio and text-domain. As pre-processing step, we created a coordinate representation that results in SSMs of the same dimensions for both domains. This allowed us an easy adaptation of an existing model to capture the complementary structure of both domains. Through experiments, we showed that the multimodal system outperforms the previous existing model based only on text.

How can we effectively control data-driven models by context information? Can we then use the prior knowledge about the audio signal defined by the lyrics to improve the isolation of the singing voice from the mixture?

We centered our effort in the task of music source separation that aims to isolate the different instruments that appear in an audio mixture. Concretely, we adapted data-driven approaches to be controlled by some external information. In this paradigm, we have access to the mixture and the isolated sources that compose it. Thus, models learn in a supervised learning way to either compute a mask to isolate the source from the background or to obtain directly the clean spectral representations.

Our approach consisted of a model-based system that uses external information to condition the behavior of a generic model. We presented a novel approach where the conditioning is implemented using FiLM, a well-known conditioning approach that comes from the image processing domain. Chapter 8 provides a first approximation to conditioned learning for music source separation. We explored the multitask source separation and used a weak conditioning system where we used the prior knowledge about the different instruments to perform several instrument separations with a single model without losing performance and adding just a small number of parameters. This showed that we can effectively control a generic neural network by some external information. In this case for performing several instrument isolations.

In Chapter 9, we extended this approach to singing vocal source separation in an informed-source separation scenario, where we aimed to use prior information about the target source to improve separation. Although previous approaches employed the prior information provided by the scores, we

explored the phonetic characteristics. Looking for combining the power of data-driven models with the adaptability of informed approaches, we used the multitrack version of DALI and the phonetic information defined in the annotations to introduce additional flexibility in our model and adapt it to the observed signals. We used this information to improve vocal separations and proposed a weak and strong conditioning strategy that outperforms the standard architecture without external information.

Overall, we found that conditioned learning was an effective method for controlling the behavior of a generic model. We showed that it is not necessary to introduce complex architectures and just with a few but precise parameters we can improve the performances of the current architectures.

10.2 Future work

There is a large number of interesting new research directions hinted at in this work. We group them according to the different topics treated in this dissertation.

10.2.1 Dataset

In this work, we created two different versions of the DALI dataset. We plan also to extend the data contained in the annotations. Currently DALI has *musical note event* labels where a note event consists of a start time, end time and pitch. This is quantized at the note level but it does not contain any information about the f_0 i.e. the frequency evolution of notes in the audio signal. This can be added by separating the vocals and tracking the f_0 in a similar way to what was done in MedleyDB. Another area of improvement is the phoneme information, which was extracted automatically from the word level. There are two directions. First of all, this process is error prone in both word annotations during the labelling and the automatic phonetization. Further metrics to estimate the quality of these annotations are needed. Secondly, the phonemes of a word are represented as a bag of phonemes that occurr at the same time without explict onsets or offsets rather than as a succession of phonemes. Providing detailed alignment for individual phonemes will benefit to the strong conditioning source separation and lead to better results. This will also be helpful for the work detailed in Section 10.2.4.

Concerning the error probability function, we believe the error detection model could be applied to scenarios other than training. A natural future work is to integrate it as a guide or objective measure to evaluate the performance of the different proposed methods for solving the local errors (see Chapter 6). This will allow the evaluation of many different alignment configurations as well as exploring ways of combining the different information (text and notes) to create a more robust error solving system.

Regarding the proposed data cleansing strategy itself, we plan to directly apply it to any kind of note event annotation (not only frame level), as well as extend it for other types of time-varying

annotations such as chords or beats. We can also use it to streamline manual annotation efforts by using the model to select time regions that are likely wrong and send them to an expert for correction. We would also like to explore how this idea can be generalized to other domains beyond music and to test the contribution of different factors including the amount of noise in a dataset and the nature of the noise.

Finally, the multitrack subset contains the instruments that define the accompaniment section. Cleaning and preparing these instruments will help in future work related to multitask source separation. Additionally, the *musical note event* labels are only employed for creating the error probability function. Scores are also one of the most important representation of music. There is a wide area of research integrating this information in the current setup.

10.2.2 Structures

We only explored a scenario in which the audio information helped to improve the structure segmentation of text information. The first and most clear extension of this work would be to use the text information to help in detecting musical structures. The hierarchical information defined by the different levels of granularity would (1) help in disambiguating many of the inconsistencies (meaning having difficulties in selecting when to segment) presented in music structures detection systems, (2) define hierarchical relationships between musical motives (lyrics lines define melodic lines) and (3) add a semantic dimension to each structure i.e. being able to label each section and analyze the topic finding correlations between the evolution of the lyrics and the music, various lyric sections define musical structures: verses reveal stories and choruses sum up the emotional message. This multimodal formalization of the structure analysis will help in improving the understanding that methods have about what a song is.

10.2.3 Source separation

The C-U-Net model was tested only for four instruments. We plan to extend it to more instruments to explore its limitations. We can achieve this after cleaning and preparing the other tracks presented in the multitrack version of DALI. We would like also to explore the performance for complex tasks i.e. separating several instruments combinations (e.g. vocals+drums). Regarding FiLM layers, we plan to explore how they affect the computation to deeply understand their behavior to use it more effectively. For instance, we would like to visualize the latent spaces and the encoder blocks to better understand where to apply the conditioning.

As we mentioned when exploring vocal isolation, the strong conditioning worked better when the basis have limited flexibility, especially when the same value was applied to all the frequency dimension. To overcome this limitation, we would like to make the bases dependent on other data such as the notes annotations defined in DALI. Additionally, we are not considering possible errors in the alignment that may lead to errors in the separation. We would like to integrate attention

mechanisms to add further flexibility to the model and be robust to these issues. As mentioned in the previous sections, once we have access to the explicit alignment per phoneme rather than per word, the system will be able to see each phoneme in isolation which will help in the performance. An interesting line of work is to adapt the model to directly obtain the phoneme activations matrix from the mixture so that it will not need it for facing the real-world scenario.

We also would like to integrate our conditioning mechanism to other architectures such as Wave-U-Net or TASNet. Likewise, we aim to develop ways of adding new conditions (namely new instrument isolation) to a trained model and find ways of separating the joint training, to create a generic model that can be easily adapted to new control mechanisms.

10.2.4 Approaching the multimodal scenario

A line of research not explored in this work is how to use DALI to create systems that automatically generate multimodal data i.e. lyrics and vocal notes aligned in time. Tasks such as automatic lyric alignment, singing to text/notes transcription can greatly benefit from this data. Recently, there have been many advances in these fields such as the use of alignment losses like Connectionist temporal classification (CTC), the improvement of f_0 methods and speech transcription methods that make us think that soon we will be able to automatically obtain aligned lyrics or notes to formulate multimodal models that do not depend on having this data.

Additionally, we can formulate tasks such as source separation to perform both tasks, improving the separation as well as learning how to create features that capture the phonetic information over time.

10.2.5 Other multimodal scenarios

Finally, there are many other MIR tasks that can benefit from the insights we can derive from such datasets. Problems like cover detection, genre classification, or mood estimation are directly connected with the lyrics. In covers, the vocal melody and the lyrics are usually some of the main elements that remain from the original song. The topics of lyrics are highly related to music genres even defining concept albums where tracks are part of a single central narrative that holds a collective meaning. Additionally, in popular music, lyrics and music work together for conveying defined emotions which make mood estimation another natural multimodal scenario to explore.

Until now we have talked only about discriminative approaches where we aim to label frames or the song as a whole. However, there is a vast range of applications that can be investigated in the generative field. For example, exploring ways of automatically generating lyrics given a particular audio melody or the other way round, to provide vocal melodies given lyrics.

10.3 Conclusions

In this dissertation, we faced music multimodal learning as a whole. We developed novel methods for automatically creating a large dataset that fitted our needs and evaluating the quality of the annotations. We presented several strategies that combined different multimodal formalizations as well as made use of the different dimensions and levels of hierarchy presented in our dataset. The body of work presented in this dissertation can be summarized as follows. We showed that data creation and model learning can work together and benefit each other. We further developed ways of dealing with nosiy data. We then saw how multimodal data-driven methods can exploit inherent relationships between domains and were able to outperform existing models based on a single modality. We have also integrated conditioning mechanisms for effectively controlling standard architectures with respect to some external information, verifying that we can control them with just a few but effective parameters.

We hope the results and methods proposed in this thesis prompt novel research into multimodal learning methods for efficiently and effectively developing new work and encourage researchers to continue in this direction.

Appendix A

Acronyms

CNN Convolutional Neural Network. (pages 19, *Glossary:* CNN)

CQT Constant-Q transform. (page *Glossary:* CQT)

C-U-Net Conditioned-U-Net. (page *Glossary:* C-U-Net)

DALI Dataset of Aligned Lyric Information. (pages 29, *Glossary:* DALI)

DFT Discrete Fourier transform. (pages 50, *Glossary:* DFT)

DNN Deep Neural Networks. (pages 16, *Glossary:* DNN)

DTW Dynamic Time Warping. (pages 70, *Glossary:* DTW)

FiLM Feature-wise Linear Modulation. (page *Glossary:* FiLM)

ISMIR International Society for Music Information Retrieval. (pages 33, *Glossary:* ISMIR)

MIR Music Information Retrieval. (pages 6, *Glossary:* MIR)

MLS Log amplitude Scale. (page *Glossary:* MLS)

NCC Normalized Cross-Correlation. (pages 50, *Glossary:* NCC)

SSM Self-Similarity Matrices. (pages 96, *Glossary:* SSM)

SVD Singing Voice Detector. (pages 41, *Glossary:* SVD)

SVP Singing Voice Probability vector. (pages 41, *Glossary:* SVP)

VAS Voice Annotation Sequence. (pages 41, *Glossary:* VAS)

WASABI Web Audio Semantic Aggregated in the Browser for Indexation. (pages 43, *Glossary:* WASABI)

Appendix B

Glossary

f_0 is a MIR task that estimates the fundamental note frequencies in polyphonic music computing a matrix over time with where each frame stores the note likelihoods. (pages 39, 66, 67, 70, 71, 73, 84, 87, 92, 93, 109, 137, 138, 140)

batch normalization is a deep neural network technique that normalizes the output of one layer before applying the activation function. (pages 26, 86, 110, 113–115)

C-U-Net our proposed conditioned architecture for source separation. It is based on a U-Net architecture and adds FiLM layer to the encoder for controlling its behavior. (pages 103, 111, 112, 114, 115, 117–121, 125, 126, 139, 143)

CNN Convolutional Neural Network is a neural network architecture in which at least one layer is a convolutional layer i.e. a convolutional filter passes along an input matrix. (pages 19, 20, 22–24, 28, 41, 50, 86, 97, 98, 106, 115, 143, 148)

conditioned learning is a machine learning paradigm where we want to process some information in the context of another. For that, we create a generic model that changes its behavior instead of having a dedicated model for each possible context information. (pages 104, 105, 116, 120, 137, 138)

CQT Constant-Q transform is a frequency transformation where the frequency bins are logarithmically spaced and with equal center frequencies-to-bandwidth ratios. (pages 71, 82, 83, 86, 143)

DALI our proposed dataset with synchronized audio, lyrics, and notes (Meseguer-Brocal et al., 2018). (pages 29, 30, 33–39, 41, 42, 46, 53, 54, 56, 57, 59–63, 65, 67, 73, 75, 76, 80–82, 84–86, 88, 90–93, 95, 98, 99, 102, 103, 120, 125, 129–132, 136–140, 143, 150)

data augmentation consists of a series of techniques that artificially enriches or "augments" the training to better approximate the real-world and prevent overfitting. (pages 26, 117, 120)

data cleansing is a machine learning strategy for cleaning erroneous labels in datasets. (pages 76, 77, 79, 80, 82, 83, 87, 91–93, 136, 138)

DFT Discrete Fourier transform is a frequency transformation that represents a signal as a sum of N complex-valued Fourier coefficients (magnitude and phase) for sinusoids of varying frequency called "frequency bins".. (pages 50, 112, 143, 147)

DNN Deep Neural Networks are machine learning algorithms organized in consecutive layers built on the output from the previous layer. They introduce a nonlinearity function between layers for model complex relationships between input and output. (pages 16–19, 22, 23, 46, 47, 49, 143)

dropout disables a random selection of a fixed number of neurons enabeling the neurons to be useful in conjunction with several random subsets of neurons. (pages 25, 26, 86, 113–115)

DTW Dynamic Time Warping is a measure of similarity of two series that warps them onto a common set of instants such that the distance between them is the smallest. It uses dynamic programming to find monotonic alignment such that the sum of a distance-like cost between aligned feature vectors is minimized. (pages 70, 99–101, 123, 143)

error probability function is a function over time that predicts if a given label is an error or not with respect to an audio segment. (pages 83, 86, 87, 89, 92, 136, 138, 139)

FiLM layers conditions the network computation by applying an affine transformation to intermediate features. (pages 103, 109–115, 117, 119–122, 137, 139, 143, 145)

fully-connected is a deep nueral network architecture where layers contain a set of connected neurons that act in parallel where each neuron is connected to all neurons in the previous layer. (pages 17, 19, 23, 50, 86, 97, 106, 114)

hidden layer is a deep nueral net intermediate layer that has as input the output of another layer and as output an intermediate features. (pages 17, 19, 23)

ISMIR The International Society for Music Information Retrieval is the world's leading research forum on processing, searching, organising and accessing music-related data. (pages 33, 143)

Jamendo is a MIR dataset with 93 creative-commons licensed music pieces annotated by voice and no-voice. (pages 48–50, 55, 57)

machine learning Machine Learning is a research discipline that designs methods for enabling computers to learn to do particular tasks without being explicitly programmed to do so. Those methods learn from a collection of data where they automatically discover the needed patterns to carry out the desired tasks. (pages 8, 13, 16, 32, 145–148)

MedleyDB is a MIR multitrack dataset with 122 songs, with mix, stems of different instruments. The dataset is annotated in melody F0 (for 108 tracks), instrument activations and genre (for all tracks). (pages 31, 55, 57, 88, 106, 138)

MIR Music Information Retrieval is an interdisciplinary research field for understanding music audio signals that combines theories, concepts and techniques from music theory, computer science, signal processing perception and cognition. (pages 6, 8–10, 23, 28–31, 34, 41, 42, 53, 87, 93, 96, 103, 104, 120–122, 135, 140, 143, 145–147)

MLS Mel log amplitude Scale is a frequency transformation that mimics the nonlinear critical bands of the human ear by applying a bank of triangular filters to the power of the spectrogram DFT. (pages 50, 143)

multimodal is a machine learning discipline that studies how to use data from different domains that observe a common phenomenon toward resolving complex tasks. (pages 1–11, 29, 33, 92, 93, 95, 102, 121, 122, 135–137, 139–141)

NCC Normalized Cross-Correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. It deals with singals that have different energy levels scaling the cross-correlation to a factor that is related to the energy of both signals. (pages 50–52, 62, 65–67, 93, 143)

residual/skip connection is a deep neural network technique that connects the output of one layer with the input of an earlier layer, so that each new layer deviate from the identity function, which still goes through the net. This leaves the outputs of the previous layers unchanged just that we could now do additional transformations. (pages 25, 27, 106, 107, 112–114, 148)

singing voice is a MIR topic that focus on the analysis of everything that is realted with the singing voice. (pages 29, 31, 46)

source separation is a machine learning task that aims to separate the different sources that appear in an audio mixture. (pages 34, 49, 71, 83, 103–105, 108, 109, 111, 112, 116, 117, 119–122, 137, 140, 145)

SSM are a MIR feature that, given a song described as a set of sequential elements $X = x1, x2, ..., xk$, it computes a similarity measure between all the possible combinations of the set, creating a matrix that directly highlights similar elements. (pages 96–102, 137, 143)

student is one of the agents of the teacher-student paradigm and it is trained using the labeled data from a teacher acquiring its knowledge by mimicking the "teacher behaviour". (pages 41, 53, 80, 148)

supervised learning is a machine learning paradigm where the machine uses labeled data to discover functions that map input-output $f(x) = \hat{y}$. (pages 10, 13–15, 24, 25, 29, 75, 76, 78, 95, 105, 137)

SVD A Singing Voice Detector is a machine learning model based on CNN that compute the SVP. (pages 41, 47, 48, 52–55, 57–59, 61, 62, 65, 93, 136, 143)

SVP Singing Voice Probability vector is a function over time $\hat{p}(t)$ extracted from a song with $\hat{p}(t) \to 1$ when there is voice and $\hat{p}(t) \to 0$ when not. (pages 41, 47, 48, 62, 143, 148)

teacher is one of the agents of the teacher-student paradigm and it is in charge of automatically labeling the training data of another system, the student. (pages 41, 53, 80, 92, 147, 148)

teacher-student paradigm is a method for acquiring knowledge between machine learning models where one systems (the teacher) transferns its knowledge to another system (the student). The teacher automatically labels the training data of the student acquiring the its knowledge by mimicking the "teacher behaviour". This is done in the context of model compression or insufficient labeled training. (pages 14, 41, 53, 54, 56, 79, 92, 93, 136, 147, 148)

U-Net is a CNN architecture with a mirror encoder/decoder based that adds residual/skip connection connections between layers at the same hierarchical level in the encoder and decoder. (pages 24, 25, 103–106, 108, 109, 111–117, 119, 120, 122, 126, 127, 130–132, 145)

VAS The Annotation Voice Sequence vector is a function over time $vas(t)$ extracted from the annotation with $vas(t) = 1$ when the annotations have voice and $vas(t) = 0$ when not. (pages 41, 47, 62, 143)

WASABI a semantic database of song metadata collected from various music databases (Meseguer-Brocal et al., 2017) `https://wasabi.i3s.unice.fr/`. (pages 43, 44, 61, 98, 143)

Appendix C

Publications and code

C.1 Publications

1. Gabriel Meseguer-Brocal, Alice Cohen-Hadria and Geoffroy Peeters. ***DALI: a large Dataset of synchronized Audio, LyrIcs and notes, automatically created using teacher-student machine learning paradigm***. In 19th International Society for Music Information Retrieval (ISMIR) Conference, 2018

2. Gabriel Meseguer-Brocal and Geoffroy Peeters. ***Conditioned-U-Net: Introducing a Control Mechanism in the U-Net for Multiple Source Separations***. In 20th International Society for Music Information Retrieval (ISMIR) Conference, 2019

3. Gabriel Meseguer-Brocal, Alice Cohen-Hadria, Geoffroy Peeters. ***Creating DALI, a large dataset of synchronized audio, lyrics, and notes***. Transactions of the International Society for Music Information Retrieval Journal (TISMIR).

4. Michael Fell E, Yaroslav Nechaev, Gabriel Meseguer-Brocal, Elena Cabrio, Fabien Gandon and Geoffroy Peeters. ***Lyrics Segmentation via Bimodal Text-audio Representation***. Natural Language Engineering.

5. Gabriel Meseguer-Brocal and Geoffroy Peeters. ***Content based singing voice source separation via strong conditioning using aligned phonemes***. In 21th International Society for Music Information Retrieval (ISMIR) Conference, 2020.

6. Gabriel Meseguer-Brocal, Rachel Bittner, Simon Duran, Brian Brost. ***Data Cleansing with Contrastive Learning for Vocal Note Event Annotations***. In 21th International Society for Music Information Retrieval (ISMIR) Conference, 2020.

C.2 Code

During the course of this thesis, a great among of tools have been developed. Most of them are publicly available at my GitHub: `https://github.com/gabolsgabs`

This includes the package for working with DALI as well as all the necessary functions to reproduce the methods from scratch. This covers preprocessing the data, the creation of the pipelines embedded in the graph for efficiently handling the data, the creation of the different architectures, the training process as well as the evaluation. The code has been developed using `Python`. The main toolbox employed is **librosa** (McFee et al., 2015) and **mir_eval** (Raffel et al., 2014). Deep neural networks have been exploited using **TensorFlow** (Abadi et al., 2016). Each project is self-contained and can be installed used `pip`.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283. 150

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016). Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675. 30

Arandjelovic, R. and Zisserman, A. (2017). Look, listen and learn. In *The IEEE International Conference on Computer Vision (ICCV)*. 5, 81

Arpit, D., Jastrzundefinedbski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and et al. (2017). A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org. 76, 77

Ashok, A., Rhinehart, N., Beainy, F., and Kitani, K. M. (2017). N2N learning: Network to network compression via policy gradient reinforcement learning. *CoRR*. 53

Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, 16(6):345–379. 1, 5

Aytar, Y., Vondrick, C., and Torralba, A. (2016). Soundnet: Learning sound representations from unlabeled video. In *Advances in neural information processing systems*, pages 892–900. 5

Balke, S., Dittmar, C., Abeßer, J., Frieler, K., Pfleiderer, M., and Müller, M. (2018). Bridging the gap: Enriching youtube videos with jazz music annotations. *Frontiers in Digital Humanities*, 5:1. 44

Ballard, D. H. (1987). Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press. 23

Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443. 1, 2, 5

Baratè, A., Ludovico, L. A., and Santucci, E. (2013). A semantics-driven approach to lyrics segmentation. In *2013 8th International Workshop on Semantic and Social Media Adaptation and Personalization*, pages 73–79. 96

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828. 2, 105, 122

Benzi, K., Defferrard, M., Vandergheynst, P., and Bresson, X. (2016). FMA: A dataset for music analysis. *CoRR*, abs/1612.01840. 31

Berenzweig, A., Ellis, D. P. W., and Lawrence, S. (2002). Using voice segments to improve artist classification of music. In *In Processings of 22th International Conference on Immersive and Interactive Audio*. 49

Berenzweig, A. L. and Ellis, D. P. W. (2001). Locating singing voice segments within music signals. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 119–122. 49

Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2016). Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442. 3, 5

Biadsy, F., Weiss, R. J., Moreno, P. J., Kanvesky, D., and Jia, Y. (2019). Parrotron: An end-to-end speech-to-speech conversion model and its applications to hearing-impaired speech and speech separation. *Proc. Interspeech 2019*, pages 4115–4119. 123

Bittner, R. and Bosch, J. J. (2019). Generalized metrics for single-f0 estimation evaluation. In *In Proceedings of 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands. 66, 88

Bittner, R., Fuentes, M., Rubinstein, D., Jansson, A., Choi, K., and Kell, T. (2019). mirdata: Software for reproducible usage of datasets. In *In Proceedings of 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands. 38

Bittner, R., Salamon, J., Tierney, M., Mauch, M., Cannam, C., and Bello, J. (2014). Medleydb: A multitrack dataset for annotation-intensive mir research. In *In Proceedings of 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan. 31, 55, 88, 106

Bittner, R. M., McFee, B., Salamon, J., Li, P., and Bello, J. P. (2017). Deep salience representations for f0 estimation in polyphonic music. In ISMIR, editor, *18th International Society for Music Information Retrieval Conference*. 47, 84, 88

Brown, J. (1991). Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89:425–. 71, 82

Bucilua, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery. 53

Cano, E., FitzGerald, D., Liutkus, A., Plumbley, M. D., and Stöter, F.-R. (2018). Musical source separation: An introduction. *IEEE Signal Processing Magazine*, 36(1):31–40. 103, 104, 105

Chan, T.-S., Yeh, T.-C., Fan, Z.-C., Chen, H.-W., Su, L., Yang, Y.-H., and Jang, J.-S. R. (2015). Vocal activity informed singing voice separation with the ikala dataset. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 718–722. 88, 125

Chandna, P., Blaauw, M., Bonada, J., and Gomez, E. (2019). A vocoder based method for singing voice extraction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 990–994. IEEE. 107

Chandna, P., Blaauw, M., Bonada, J., and Gómez, E. (2020). Content based singing voice extraction from a musical mixture. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 781–785. IEEE. 122, 125

Chandna, P., Miron, M., Janer, J., and Gómez, E. (2017). Monoaural audio source separation using deep convolutional neural networks. In *Proc. of LVA/ICA (International Conference on Latent Variable Analysis and Signal Separation)*, Grenoble, France. 106, 125

Chazan, S. E., Gannot, S., and Goldberger, J. (2016). A phoneme-based pre-training approach for deep neural network with application to speech enhancement. In *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 1–5. IEEE. 123

Chen, L., Srivastava, S., Duan, Z., and Xu, C. (2017). Deep cross-modal audio-visual generation. *CoRR*, abs/1704.08292. 7

Cheng, H.-T., Yang, Y.-H., Lin, Y.-C., and Chen, H. H. (2009a). Multimodal structure segmentation and analysis of music using audio and textual information. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1677–1680. IEEE. 7

Cheng, H. T., Yang, Y. H., Lin, Y. C., and Chen, H. H. (2009b). Multimodal structure segmentation and analysis of music using audio and textual information. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1677–1680. 95

Chung, J. S., Senior, A., Vinyals, O., and Zisserman, A. (2017). Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3444–3453. IEEE. 5

Cohen-Hadria, A., Roebel, A., and Peeters, G. (2019). Improving singing voice separation using deep u-net and wave-u-net with data augmentation. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. 120

Cont, A., Schwarz, D., Schnell, N., and Raphael, C. (2007). Evaluation of Real-Time Audio-to-Score Alignment. In *In Proceedings of 8th International Society for Music Information Retrieval Conference*, Vienna, Austria. 46

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 26

Cui, J., Kingsbury, B., Ramabhadran, B., Saon, G., Sercu, T., Audhkhasi, K., Sethy, A., Nussbaum-Thom, M., and Rosenberg, A. (2017). Knowledge distillation across ensembles of multilingual models for low-resource languages. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. 53

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *ACOUSTICS, SPEECH AND SIGNAL PROCESSING, IEEE TRANSACTIONS ON*, pages 357–366. 99

de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. (2017). Modulating early visual processing by language. In *Proc. of NIPS (Annual Conference on Neural Information Processing Systems)*, Long Beach, CA, USA. 104, 111, 114

Défossez, A., Usunier, N., Bottou, L., and Bach, F. (2019). Demucs: Deep extractor for music sources with extra unlabeled data remixed. *arXiv preprint arXiv:1909.01174*. 107

Demetriou, A., Jansson, A., Kumar, A., and Bittner, R. M. (2018). Vocals in music matter: the relevance of vocals in the minds of listeners. In *In Proceedings of 19th International Society for Music Information Retrieval Conference*, Paris, France. 8, 122, 125

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. 30

Dinesh, K., Li, B., Liu, X., Duan, Z., and Sharma, G. (2017). Visually informed multi-pitch analysis of string ensembles. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3021–3025. IEEE. 7

Donahue, C., Henry Mao, H., and McAuley, J. (2018). The nes music database: A multi-instrumental dataset with expressive performance attributes. In *In Proceedings of 19th International Society for Music Information Retrieval Conference*, Paris, France. 31

Doras, G., Esling, P., and Peeters, G. (2019). On the use of u-net for dominant melody estimation in polyphonic music. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 66–70. 66

Duan, Z. and Pardo, B. (2011). Soundprism: An online system for score-informed source separation of music audio. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1205–1215. 124

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. (2018). Feature-wise transformations. *Distill*. https://distill.pub/2018/feature-wise-transformations. 109, 123

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. cite arxiv:1603.07285. 20

Durrieu, J.-L., Richard, G., and David, B. (2008). Singer melody extraction in polyphonic signals using source separation methods. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 169–172. IEEE. 109

Dzhambazov, G. (2017). *Knowledge-based Probabilistic Modeling for Tracking Lyrics in Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra. 32, 33, 45, 46

Emiya, V., Badeau, R., and David, B. (2009). Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654. 31

Essid, S. and Richard, G. (2012). Fusion of multimodal information in music content analysis. In *Dagstuhl Follow-Ups*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 5, 7, 135

Ewert, S. and Müller, M. (2012). Using score-informed constraints for nmf-based source separation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 129–132. IEEE. 124

Ewert, S., Pardo, B., Müller, M., and Plumbley, M. D. (2014). Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3):116–124. 7, 122, 124

Ewert, S. and Sandler, M. B. (2017). Structured dropout for weak label and multi-instance learning and its application to score-informed source separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2277–2281. IEEE. 125

Fell, M., Nechaev, Y., Cabrio, E., and Gandon, F. (2018). Lyrics segmentation: Textual macrostructure detection using convolutions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2044–2054. 96, 97, 98, 101

Fonseca, E., Pons, J., Favory, X., Font, F., Bogdanov, D., Ferraro, A., Oramas, S., Porter, A., and Serra, X. (2017). Freesound datasets: A platform for the creation of open audio datasets. In *In Proceedings of 18th International Society for Music Information Retrieval Conference*, Suzhou, China. 31

Foote, J. (2000). Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE. 96

Frenay, B. and Verleysen, M. (2014). Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869. 76, 77

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York. 78

Fritsch, J. and Plumbley, M. D. (2013). Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 888–891. IEEE. 124

Fujihara, H. and Goto, M. (2012). Lyrics-to-Audio Alignment and its Application. In *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 23–36. Dagstuhl, Germany. 31, 45

Fujihara, H., Goto, M., Ogata, J., and Okuno, H. G. (2011). Lyricsynchronizer: Automatic synchronization system between musical audio signals and lyrics. 5(6):1252–1261. 33, 45, 49

Fujishima, T. (1999). Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*. Michigan Publishing. 99

Fürniss, S. and Castellengo, M. (2016). Ecoute musicale et acoustique. *Cahiers d'ethnomusicologie. Anciennement Cahiers de musiques traditionnelles*, (29):223–226. 31

Gillet, O., Essid, S., and Richard, G. (2007). On the correlation of automatic audio and visual segmentations of music videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(3):347–355. 7

Gillet, O. and Richard, G. (2005). Automatic transcription of drum sequences using audiovisual features. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, pages iii–205. IEEE. 7

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR. 19

Goldberger, J. and Ben-Reuven, E. (2017). Training deep neural-networks using a noise adaptation layer. In *ICLR*. 78

Goldsmith, J. (1976). *Autosegmental phonology*. PhD thesis, MIT Press London. 122

Gómez, E., Blaauw, M., Bonada, J., Chandna, P., and Cuesta, H. (2018). Deep learning for singing processing: Achievements, challenges and impact on singers and listeners. *arXiv preprint arXiv:1807.03046*. 122

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 13, 17, 26

Goto, M. (2014). Singing information processing. In *ICSP*, pages 2431–2438. 31

Goto, M., Hashiguchi, H., Nishimura, T., and Oka, R. (2002). Rwc music database: Popular, classical, and jazz music databases. In *In Proc. 3rd International Conference on Music Information Retrieval*, pages 287–288. 49

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery. 5, 68

Gregorio, J. and Kim, Y. (2016). Phrase-level audio segmentation of jazz improvisations informed by symbolic data. In *ISMIR*, pages 482–487. 7

Gulluni, S., Essid, S., Buisson, O., and Richard, G. (2011). An interactive system for electro-acoustic music analysis. In *ISMIR*, pages 145–150. 6, 7

Gupta, C., Tong, R., Li, H., and Wang, Y. (2018). Semi-supervised lyrics and solo-singing alignment. In *In Proceedings of 19th International Society for Music Information Retrieval Conference*. 31, 32, 33, 46

Gupta, C., Yılmaz, E., and Li, H. (2019). Acoustic modeling for automatic lyrics-to-audio alignment. *arXiv preprint arXiv:1906.10369*. 46

Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In Bengio, S.,

Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 8527–8537. Curran Associates, Inc. 80

Hansen, J. K. (2012). Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients. In *In Proceedings of 9th Sound and Music Computing Conference*, Copenhagen, Denmark. 32

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., and Eck, D. (2018). Onsets and frames: Dual-objective piano transcription. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France. 107

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations (ICLR)*. 31

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. 27

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. (2018). Using trusted data to train deep networks on labels corrupted by severe noise. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 10456–10465. Curran Associates, Inc. 78

Higgins, I., Sonnerat, N., Matthey, L., Pal, A., Burgess, C. P., Botvinick, M., Hassabis, D., and Lerchner, A. (2017). Scan: Learning abstract hierarchical compositional visual concepts. 5

Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*. 41, 53, 79

Huang, C. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., Dai, A. M., Hoffman, M. D., and Eck, D. (2018). An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281. 107

Huang, M., Rong, W., Arjannikov, T., Jiang, N., and Xiong, Z. (2016). *Bi-Modal Deep Boltzmann Machine Based Musical Emotion Classification*, pages 199–207. Springer International Publishing, Cham. 7

Huang, P., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. (2014). Deep learning for monaural speech separation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1562–1566. 106

Huang, P., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. (2015). Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 23(12). 106

Humphrey, E. J., Montecchio, N., Bittner, R., Jansson, A., and Jehan, T. (2017). Mining labeled data from web-scale collections for vocal activity detection in music. In *In Proceedings of 18th International Society for Music Information Retrieval Conference*, Suzhou, China. 31, 50

Humphrey, E. J., Reddy, S., Seetharaman, P., Kumar, A., Bittner, R. M., Demetriou, A., Gulati, S., Jansson, A., Jehan, T., Lehner, B., et al. (2018). An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music. *IEEE Signal Processing Magazine*, 36(1):82–94. 8, 122

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org. 26, 114

Iskandar, D., Wang, Y., Kan, M.-Y., and Li, H. (2006). Syllabic level automatic synchronization of music signals and text lyrics. In *Proceedings of the 14th ACM International Conference on Multimedia*, MM '06, pages 659–662, New York, NY, USA. ACM. 32, 33, 45, 46

Jansson, A., Bittner, R. M., Ewert, S., and Weyde, T. (2019). Joint singing voice separation and f0 estimation with deep u-net architectures. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE. 109

Jansson, A., Humphrey, E. J., Montecchio, N., Bittner, R., Kumar, A., and Weyde, T. (2017). Singing voice separation with deep u-net convolutional networks. In *In Proceedings of 18th International Society for Music Information Retrieval Conference*, Suzhou, China. 71, 83, 104, 106, 111, 112, 113, 116, 118, 126

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*. 80

Kadandale, V. S., Montesinos, J. F., Haro, G., and Gómez, E. (2020). Multi-task u-net for music source separation. *arXiv preprint arXiv:2003.10414*. 108

Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., and Uszkoreit, J. (2017). One model to learn them all. *arXiv preprint arXiv:1706.05137*. 5

Kan, M.-Y., Wang, Y., Iskandar, D., Nwe, T. L., and Shenoy, A. (2008). Lyrically: Automatic synchronization of textual lyrics to acoustic music signals. 16(2):338 – 349. 32, 33, 45

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA. IEEE Computer Society. 5

Kavalerov, I., Wisdom, S., Erdogan, H., Patton, B., Wilson, K., Le Roux, J., and Hershey, J. R. (2019). Universal sound separation. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 175–179. IEEE. 108

Kim, T., Song, I., and Bengio, Y. (2017). Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. *CoRR*, abs/1707.06065. 107, 114

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 24, 113

Kinoshita, K., Delcroix, M., Ogawa, A., and Nakatani, T. (2015). Text-informed speech enhancement with deep neural networks. In *Sixteenth Annual Conference of the International Speech Communication Association*. 121, 123

Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539. 5

Krause, J., Sapp, B., Howard, A., Zhou, H., Toshev, A., Duerig, T., Philbin, J., and Fei-Fei, L. (2016). The unreasonable effectiveness of noisy data for fine-grained recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 301–320, Cham. Springer International Publishing. 30, 80

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. 30

Kruspe, A. M. (2016). Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing. In *In Proceedings of 17th International Society for Music Information Retrieval Conference*, pages 358–364, New York City, United States. 31, 32, 33, 45

Ladd, D. R. (2008). *Intonational phonology*. Cambridge University Press. 122

Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations (ICLR)*. 76, 77, 79

Laurier, C., Grivolla, J., and Herrera, P. (2008). Multimodal music mood classification using audio and lyrics. In *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, ICMLA '08, pages 688–693, Washington, DC, USA. IEEE Computer Society. 6

Le Cun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324. 30

Lea, C., Vidal, R., Reiter, A., and Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, pages 47–54. Springer. 107

LeCun, Y. and Bengio, Y. (1995). *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA. 19

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562. 127

Lee, J. H., Choi, H.-S., and Lee, K. (2019). Audio query-based music source separation. In ISMIR, editor, *20th International Society for Music Information Retrieval Conference*. 108

Lee, K., Choi, K., and Nam, J. (2018a). Revisiting singing voice detection: a quantitative review and the future outlook. 50

Lee, K. and Cremer, M. (2008). Segmentation-based lyrics-audio alignment using dynamic programming. In *In Proceedings of 9th International Society for Music Information Retrieval Conference*, Drexel University in Philadelphia, PA USA. 46

Lee, K.-H., He, X., Zhang, L., and Yang, L. (2018b). Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 81

Lee, S. W. and Scott, J. (2017). Word level lyrics-audio synchronization using separated vocals. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 32

Leglaive, S., Hennequin, R., and Badeau, R. (2015). Singing voice detection with deep recurrent neural networks. In IEEE, editor, *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, pages 121–125, Brisbane, Australia. 49

Lehner, B., Widmer, G., and Bock, S. (2015). A low-latency, real-time-capable singing voice detection method with lstm recurrent neural networks. In *2015 23rd European Signal Processing Conference (EUSIPCO)*. 49

LeMagoarou, L., Ozerov, A., and Duong, N. Q. K. (2015). Text-informed audio source separation. example-based approach using non-negative matrix partial co-factorization. *Journal of Signal Processing Systems*, 79(2):117–131. 123

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. 97

Li, J., Zhao, R., Huang, J.-T., and Gong, Y. (2014). Learning small-size dnn with output-distribution-based criteria. In *Interspeech*. 41

Liem, C. C., Müller, M., Eck, D., Tzanetakis, G., and Hanjalic, A. (2011). The need for music information retrieval with user-centered and multimodal strategies. In *Proceedings of the 1st*

International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies, MIRUM '11, pages 1–6, New York, NY, USA. ACM. 6

Liutkus, A., Durrieu, J.-L., Daudet, L., and Richard, G. (2013). An overview of informed audio source separation. In *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4. IEEE. 123

Liutkus, A., Pinel, J., Badeau, R., Girin, L., and Richard, G. (2012). Informed source separation through spectrogram coding and data embedding. *Signal Processing*, 92(8):1937–1949. 108

Lu, X., Tsao, Y., Matsuda, S., and Hori, C. (2013). Speech enhancement based on deep denoising autoencoder. In *INTERSPEECH*. 24

Luo, Y. and Mesgarani, N. (2019). Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266. 107, 124, 132

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *The European Conference on Computer Vision (ECCV)*. 76

Mahedero, J. P. G., Martínez, A., Cano, P., Koppenberger, M., and Gouyon, F. (2005). Natural language processing of lyrics. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 475–478, New York, NY, USA. ACM. 96

Maia, L., Fuentes, M., Biscainho, L., Rocamora, M., and Essid, S. (2019). Sambaset: A dataset of historical samba de enredo recordings for computational music analysis. In *In Proceedings of 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands. 31

Malach, E. and Shalev-Shwartz, S. (2017). Decoupling "when to update" from "how to update". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 961–971, Red Hook, NY, USA. Curran Associates Inc. 80

Manilow, E., Seetharaman, P., and Pardo, B. (2019). Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments. *arXiv preprint arXiv:1910.12621*. 109

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In Honkela, T., Duch, W., Girolami, M., and Kaski, S., editors, *Artificial Neural Networks and Machine Learning – ICANN 2011*, pages 52–59, Berlin, Heidelberg. Springer Berlin Heidelberg. 24

Mauch, M., Cannam, C., Bittner, R. M., Fazekas, G., Salamon, J., Dai, J., Bello, J. P., and Dixon, S. (2015). Computer-aided melody note transcription using the Tony software: Accuracy and efficiency. In *International Conference on Technologies for Music Notation and Representation (TENOR)*. 31

Mauch, M. and Ewert, S. (2013). The audio degradation toolbox and its application to robustness evaluation. In ISMIR, editor, *18th International Society for Music Information Retrieval Conference*. 26

Mauch, M., Fujihara, H., and Goto, M. (2010). Lyrics-to-audio alignment and phrase-level segmentation using incomplete internet-style chord annotations. In *7th Sound and Music Computing Conference*, Barcelona, Spain. SMC. 7

Mauch, M., Fujihara, H., and Goto, M. (2012). Integrating additional chord information into hmm-based lyrics-to-audio alignment. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):200–210. 7, 32

Mauch, M., Fujihara, H., Yoshii, K., and Goto, M. (2011). Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *In Proceedings of 12th International Society for Music Information Retrieval Conference*, pages 233–238, Miami, USA. 46, 49

Mayer, F., Williamson, D., Mowlaee, P., and Wang, D. (2017). Impact of phase estimation on single-channel speech separation based on time-frequency masking. *The Journal of the Acoustical Society of America*, 141:4668–4679. 116

Mayer, R., Neumayer, R., and Rauber, A. (2008). Combination of audio and lyrics features for genre classification in digital audio collections. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, pages 159–168, New York, NY, USA. ACM. 6

McFee, B. and Lanckriet, G. (2011). Learning multi-modal similarity. *Journal of machine learning research*, 12(Feb):491–523. 7

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8. 150

McFee, B., Salamon, J., and Bello, J. P. (2018). Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2180–2193. 127

McGuinness, K., Gillet, O., O'Connor, N. E., and Richard, G. (2007). Visual analysis for drum sequence transcription. In *2007 15th European Signal Processing Conference*, pages 312–316. IEEE. 7

McVicar, M., Ni, Y., Santos-Rodriguez, R., and Bie, T. D. (2011). Using online chord databases to enhance chord recognition. *Journal of New Music Research*, 40(2):139–152. 7

Mesaros, A. (2013). Singing voice identification and lyrics transcription for music information retrieval invited paper. In *7th Conference on Speech Technology and Human - Computer Dialogue (SpeD)*, pages 1–10. 31, 49

Mesaros, A. and Virtanen, T. (2010). Automatic recognition of lyrics in singing. 2010:1–11. 32, 33, 45

Meseguer-Brocal, G., Bittner, R., Durand, S., and Brost, B. (2020a). Data cleansing with contrastive learning for vocal note event annotations. In ISMIR, editor, *21st International Society for Music Information Retrieval Conference*. 81

Meseguer-Brocal, G., Cohen-Hadria, A., and Peeters, G. (2018). Dali: a large dataset of synchronised audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *In Proceedings of 19th International Society for Music Information Retrieval Conference*, Paris, France. 29, 31, 32, 33, 35, 54, 57, 88, 90, 145

Meseguer-Brocal, G., Cohen-Hadria, A., and Peeters, G. (2020b). Creating dali, a large dataset of synchronized audio, lyrics, and notes. *Transactions of the International Society for Music Information Retrieval*. 57, 86, 91

Meseguer-Brocal, G. and Peeters, G. (2019). Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Delft, Netherlands. 122, 123, 124, 125, 126, 127, 128

Meseguer-Brocal, G. and Peeters, G. (2020). Content based singing voice source separation via strong conditioning using aligned phonemes. In ISMIR, editor, *21st International Society for Music Information Retrieval Conference*. 127

Meseguer-Brocal, G., Peeters, G., Pellerin, G., Buffa, M., Cabrio, E., Faron Zucker, C., Giboin, A., Mirbel, I., Hennequin, R., Moussallam, M., Piccoli, F., and Fillon, T. (2017). WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conf.*, London, U.K. Queen Mary University of London. 43, 148

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, page 1003–1011, USA. 30, 79

Miron, M., Carabias-Orti, J. J., Bosch, J. J., Gómez, E., and Janer, J. (2016). Score-informed source separation for multichannel orchestral recordings. *Journal of Electrical and Computer Engineering*, 2016. 124

Miron, M., Carabias Orti, J. J., and Janer Mestres, J. (2015). Improving score-informed source separation for classical music through note refinement. In *Müller M, Wiering F, editors. Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference; 2015 Oct 26-30; Málaga, Spain. Canada: International Society for Music Information Retrieval; 2015.* International Society for Music Information Retrieval (ISMIR). 124

Miron, M., Janer Mestres, J., and Gómez Gutiérrez, E. (2017). Monaural score-informed source separation for classical music using convolutional neural networks. In *Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017. 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Canada]: ISMIR; 2017. p. 55-62.* International Society for Music Information Retrieval (ISMIR). 121, 122, 124, 125

Mishkin, D., Sergievskiy, N., and Matas, J. (2017). Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*. 76, 77

Mnih, V. and Hinton, G. (2012). Learning to label aerial images from noisy data. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 203–210, Madison, WI, USA. 30, 79

Müller, M., Kurth, F., Damm, D., Fremerey, C., and Clausen, M. (2007). *Lyrics-Based Audio Retrieval and Multimodal Navigation in Music Collections*, pages 112–123. Berlin, Heidelberg. 7, 32, 33, 45, 46

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress. 19

Nakano, T., Koyama, Y., Hamasaki, M., and Goto, M. (2020). Interactive deep singing-voice separation based on human-in-the-loop adaptation. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 78–82. 109

Nakano, T., Yoshii, K., Wu, Y., Nishikimi, R., Lin, K. W. E., and Goto, M. (2019). Joint singing pitch estimation and voice separation based on a neural harmonic structure renderer. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 160–164. IEEE. 109

Nettleton, D. F., Orriols-Puig, A., and Fornells, A. (2010). A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33:275–306. 76, 77

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. (2011). *Multimodal deep learning*, pages 689–696. 5

Nieto, O., McCallum, M., Davies, M., Robertson, A., Stark, A., and Egozy, E. (2019). The harmonix set: Beats, downbeats, and functional segment annotations of western popular music. In *In Proceedings of 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands. 31

Nugraha, A. A., Liutkus, A., and Vincent, E. (2016a). Multichannel audio source separation with deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1652–1664. 106

Nugraha, A. A., Liutkus, A., and Vincent, E. (2016b). Multichannel music separation with deep neural networks. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1748–1752. IEEE. 106

Oramas, S., Nieto, O., Barbieri, F., and Serra, X. (2017a). Multi-label music genre classification from audio, text, and images using deep features. *CoRR*. 6

Oramas, S., Nieto, O., Sordo, M., and Serra, X. (2017b). A deep multimodal approach for cold-start music recommendation. *CoRR*. 7

Pardo, B., Liutkus, A., Duan, Z., and Richard, G. (2018). Applying source separation to music. 104, 105

Parvaix, M. and Girin, L. (2010). Informed source separation of linear instantaneous under-determined audio mixtures by source index embedding. *IEEE Transactions on audio, speech, and language processing*, 19(6):1721–1733. 108

Parvaix, M., Girin, L., and Brossier, J.-M. (2009). A watermarking-based method for informed source separation of audio signals with a single sensor. *IEEE Transactions on audio, speech, and language processing*, 18(6):1464–1475. 108

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. (2017). Making deep neural networks robust to label noise: A loss correction approach. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

Peeters, G. and Fort, K. (2012). Towards a (better) Definition of Annotated MIR Corpora. In *In Proceedings of 13th International Society for Music Information Retrieval Conference*, Porto, Portugal. 38

Peeters, G. and Rodet, X. (2004). A large set of audio feature for sound description (similarity and classification) in the cuidado project. Technical report, Ircam, Analysis/Synthesis Team, 1 pl. Igor Stravinsky, 75004 Paris, France. 97

Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *ArXiv*, abs/1712.04621. 26, 104, 105, 109, 110, 111, 114, 117, 123, 127

Prétet, L., Hennequin, R., Royo-Letelier, J., and Vaglio, A. (2019). Singing voice separation: A study on training data. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 506–510. IEEE. 130

Raffel, C. (2016). Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. 31, 46, 82

Raffel, C., Mcfee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., and Ellis, D. P. W. (2014). mir_eval: a transparent implementation of common mir metrics. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Porto, Portugal. 116, 129, 150

Rafii, Z., Liutkus, A., Stöter, F.-R., Ioannis Mimilakis, S., Fitzgerald, D., and Pardo, B. (2018). An Overview of Lead and Accompaniment Separation in Music. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 26(8). 104, 105

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., and Bittner, R. (2017). The MUSDB18 corpus for music separation. https://zenodo.org/record/1117372. 106, 115, 130

Ramachandram, D. and Taylor, G. W. (2017). Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108. 1, 5

Ramona, M., Richard, G., and David, B. (2008). Vocal detection in music with support vector machines. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*. 32, 48, 55

Reed, S. E., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *CoRR*, abs/1412.6596. 76, 78, 80

Regnier, L. and Peeters, G. (2009). Singing Voice Detection in Music Tracks using Direct Voice Vibrato Detection. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, page 1, taipei, Taiwan. 49

Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to reweight examples for robust deep learning. In *ICML*. 80

Rivest, R. (1992). The md5 message-digest algorithm. 38

Rocamora, M. and Herrera, P. (2007). Comparing audio descriptors for singing voice detection in music audio files. In *Brazilian symposium on computer music, 11th. san pablo, brazil*, volume 26, page 27. 48

Rodriguez-Serrano, F. J., Duan, Z., Vera-Candeas, P., Pardo, B., and Carabias-Orti, J. J. (2015). Online score-informed source separation with adaptive instrument models. *Journal of New Music Research*, 44(2):83–96. 124

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing. 24, 25, 112

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536. 17

Ryan, M. S. and Nudd, G. R. (1993). The viterbi algorithm. Technical report, GBR. 69

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49. 100

Salamon, J. and Gómez, E. (2012). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770. 47

Salamon, J., Gómez, E., Ellis, D. P. W., and Richard, G. (2014). Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *Signal Processing Magazine*, 31(2):118–134. 88

Samuel, D., Ganeshan, A., and Naradowsky, J. (2020). Meta-learning extractors for music source separation. *arXiv preprint arXiv:2002.07016*. 108, 122, 124

Schlüter, J. (2016). Learning to pinpoint singing voice from weakly labeled examples. In *In Proceedings of 17th International Society for Music Information Retrieval Conference*, New York City, USA. 50

Schlüter, J. and Grill, T. (2015). Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *In Proceedings of 16th International Society for Music Information Retrieval Conference*, Malaga, Spain. 50

Schulze-Forster, K., Doire, C., Richard, G., and Badeau, R. (2019). Weakly informed audio source separation. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 273–277. IEEE. 122, 125, 129

Schulze-Forster, K., Doire, C. S., Richard, G., and Badeau, R. (2020). Joint phoneme alignment and text-informed speech separation on highly corrupted speech. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7274–7278. IEEE. 124, 132

Seetharaman, P., Wichern, G., Venkataramani, S., and Le Roux, J. (2019). Class-conditional embeddings for music source separation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 301–305. IEEE. 108, 122, 124

Settles, B. (2008). *Curious Machines: Active Learning with Structured Instances Table of Contents*. PhD thesis, Stanford Music Department. 30, 80

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R. J., Saurous, R. A., Agiomyrgiannakis, Y., and Wu, Y. (2018). Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, Calgary, Canada. 107, 115

Sheng Guo, Weilin Huang, H. Z. C. Z. D. D. M. R. S. and Huang, D. (2018). Curriculumnet: Weakly supervised learning from large-scale web images. In *European Conference on Computer Vision (ECCV)*. 79

Simonetta, F., Ntalampiras, S., and Avanzini, F. (2019). Multimodal music information processing and retrieval: Survey and future challenges. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 10–18. IEEE. 6, 7, 135

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556. 23

Simpson, A. J. R., Roma, G., and Plumbley, M. D. (2015). Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. abs/1504.04658. 49

Slizovskaia, O., Haro, G., and Gómez, E. (2020). Conditioned source separation for music instrument performances. *arXiv preprint arXiv:2004.03873*. 122, 123, 124, 128, 129

Slizovskaia, O., Kim, L., Haro, G., and Gomez, E. (2019). End-to-end sound source separation conditioned on instrument labels. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 306–310. IEEE. 108, 122, 124

Smith, J. (2015). *Correlation Analyses of Encoded Music Performance*. PhD thesis, Stanford Music Department. 31, 32

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218. 5

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*. 78

Soulez, F., Rodet, X., and Schwarz, D. (2003). Improving polyphonic and poly-instrumental music to score alignment. In *In Proceedings of 4th International Society for Music Information Retrieval Conference*, page 6, Baltimore, United States. 46

Srivastava, N. and Salakhutdinov, R. (2012). Learning representations for multimodal data with deep belief nets. 5

Srivastava, N. and Salakhutdinov, R. (2014). Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15:2949–2980. 5, 25

Stoller, D., Durand, S., and Ewert, S. (2019). End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, pages 5275–5279. 31, 32, 46, 72, 133

Stoller, D., Ewert, S., and Dixon, S. (2018a). Adversarial semi-supervised audio source separation applied to singing voice extraction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2391–2395. IEEE. 107

Stoller, D., Ewert, S., and Dixon, S. (2018b). Jointly detecting and separating singing voice: A multi-task approach. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 329–339. Springer. 109

Stoller, D., Ewert, S., and Dixon, S. (2018c). Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proc. of ISMIR (International Society for Music Information Retrieval)*, Paris, France. 104, 107, 111, 118, 126

Stoter, F.-R., Uhlich, S., Liutkus, A., and Mitsufuji, Y. (2019). Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*. 106

Strub, F., Seurin, M., Perez, E., de Vries, H., Mary, J., Preux, P., Courville, A. C., and Pietquin, O. (2018). Visual reasoning with multi-hop feature modulation. In *Proc. of ECCV (European Conference on Computer Vision)*, Munich, Germany. 104, 111

Su, F. and Xue, H. (2017). Graph-based multimodal music mood classification in discriminative latent space. In *MultiMedia Modeling - 23rd International Conference, MMM 2017, Reykjavik, Iceland, January 4-6, 2017, Proceedings, Part I*, pages 152–163. 7

Su, L. and Yang, Y.-H. (2015). Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*. 31

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L. D., and Fergus, R. (2015). Training convolutional networks with noisy labels. In ICLR, editor, *3rd International Conference on Learning Representations Conference*. 76, 77

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *The IEEE International Conference on Computer Vision (ICCV)*. 9, 30

Tachibana, H., Ono, T., Ono, N., and Sagayama, S. (2010). Melody line estimation in homophonic music audio signals based on temporal-variability of melodic source. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 425–428. 49

Takahashi, N., Singh, M. K., Basak, S., Sudarsanam, P., Ganapathy, S., and Mitsufuji, Y. (2020). Improving voice separation by incorporating end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 41–45. IEEE. 124, 132

Thickstun, J., Harchaoui, Z., and Kakade, S. M. (2017). Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*. 31

Tokozume, Y., Ushiku, Y., and Harada, T. (2018). Between-class learning for image classification. In *Proc. of CVPR (Conference on Computer Vision and Pattern Recognition)*, Salt Lake City, UT, USA. 117

Tzinis, E., Wisdom, S., Hershey, J. R., Jansen, A., and Ellis, D. P. (2019). Improving universal sound separation using sound classification. *arXiv preprint arXiv:1911.07951*. 108, 122

Uhlich, S., Giron, F., and Mitsufuji, Y. (2015). Deep neural network based instrument extraction from music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2135–2139. 106

Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., and Mitsufuji, Y. (2017). Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265. IEEE. 106

Vaglio, A., Hennequin, R., Moussallam, M., Richard, G., and d'Alché Buc, F. (2020). Audio-based detection of explicit content in music. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 526–530. IEEE. 133

Vahdat, A. (2017). Toward robustness against label noise in training deep discriminative neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5596–5605. Curran Associates, Inc. 78

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499. 5, 107

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2017). Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433. 107

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., and Saenko, K. (2014). Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*. 5

Vincent, E., Araki, S., and Bofill, P. (2009). The 2008 Signal Separation Evaluation Campaign: A community-based approach to large-scale evaluation. In *8th Int. Conf. on Independent Component Analysis and Signal Separation (ICA)*, pages 734–741, Paraty, Brazil. 106

Vincent, E., Gribonval, R., and Févotte, C. (2006). Performance measurement in blind audio source separation. *IEEE/ACM TASLP (Transactions on Audio Speech and Language Processing)*, 14(4). 116, 129

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5

Vinyes, M. (2008). MTG MASS database. `http://www.mtg.upf.edu/static/mass/resources`. 106

Wang, D. and Chen, J. (2018). Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726. 105

Wang, Y., Narayanan, A., and Wang, D. (2014). On training targets for supervised speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1849–1858. 106

Wang, Z.-q., Le Roux, J., Wang, D., and Hershey, J. R. (2018). End-to-end speech separation with unfolded iterative phase reconstruction. *ArXiv*, abs/1804.10204. 106

Wang, Z.-Q., Zhao, Y., and Wang, D. (2016). Phoneme-specific speech separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 146–150. IEEE. 123

Watanabe, K., Matsubayashi, Y., Orita, N., Okazaki, N., Inui, K., Fukayama, S., Nakano, T., Smith, J., and Goto, M. (2016). Modeling discourse segments in lyrics using repeated patterns. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1959–1969. 95, 96, 97, 98

Watanabe, S., Hori, T., Le Roux, J., and Hershey, J. (2017). Student-teacher network learning with enhanced features. In *Proc. of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, pages 5275–5279. 53

Wong, C. H., Szeto, W. M., and Wong, K. H. (2007). Automatic lyrics alignment for cantonese popular music. 12(4):307–323. 32, 33, 45

Wu, C. and Lerch, A. (2017). Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *In Proceedings of 18th International Society for Music Information Retrieval Conference*, Suzhou, China. 53

Xi, Q., Bittner, R. M., Ye, X., Pauwels, J., and Bello, J. P. (2018). GuitarSet: A dataset for guitar transcription. In *In Proceedings of the 19th International Society for Music Information Retrieval (ISMIR)*. 31

Xiao, T., Xia, T., Yang, Y., Huang, C., and Wang, X. (2015). Learning from massive noisy labeled data for image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 30, 76, 79

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044. 5

Xue, H., Xue, L., and Su, F. (2015). Multimodal music mood classification by fusion of audio and lyrics. In *MultiMedia Modeling - 21st International Conference, MMM 2015, Sydney, NSW, Australia, January 5-7, 2015, Proceedings, Part II*, pages 26–37. 7

Yang, L., Chou, S., and Yang, Y. (2017). Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions. *CoRR*, abs/1703.10847. 107

Yesiler, F., Tralie, C., Correya, A., Furtado Silva, D., Tovstogan, P., Gomez, E., and Serra, X. (2019). Da-tacos: A dataset for cover song identification and understanding. In *In Proceedings of 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands. 31

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535. 21

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530. 76, 77

Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8792–8802, Red Hook, NY, USA. Curran Associates Inc. 78

Zhu, X. (2005). Semi-supervised learning literature survey. 30, 78

Zhu, Y., Chen, K., and Sun, Q. (2005). Multimodal content-based structure analysis of karaoke music. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 638–647. 7

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724. 5

Zhuang, F., Cheng, X., Luo, P., Pan, S. J., and He, Q. (2015). Supervised representation learning: Transfer learning with deep autoencoders. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 4119–4125. AAAI Press. 24