# Data-Driven Musical Version Identification: Accuracy, Scalability, and Bias Perspectives

M. Furkan Yesiler

TESI DOCTORAL UPF / 2021

Thesis Directors:

Dr. Emilia Gómez Gutierrez
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona, Spain

Dr. Joan Serrà Julià
Dolby Laboratories, Barcelona, Spain

Universitat Pompeu Fabra
Barcelona

Dissertation submitted to the Department of Information and Communication Technologies of Universitat Pompeu Fabra in partial fulfillment of the requirements for the degree of

DOCTOR PER LA UNIVERSITAT POMPEU FABRA

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

This dissertation has been carried out at the Music Technology Group of Universitat Pompeu Fabra in Barcelona, Spain, from Sep. 2018 to Nov. 2021. It has been supervised by Dr. Emilia Gómez Gutierrez and Dr. Joan Serrà Julià. This research has been supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068.

# Acknowledgments

After scribbling and erasing words that couldn't find their way to a complete sentence for the better part of an hour, I have finally realized that my brain has completely wired itself for "academic writing," and coming up with something informal and heartfelt is going to be rather difficult. So, bear with me while I express my sincere gratitude to the wonderful human beings with whom I have had the utmost pleasure of sharing joyful moments, memorable experiences, and horizon-widening ideas throughout my journey here in Barcelona, one of the most marvelous cities in the world.

First and foremost, my supervisors Joan and Emilia. I cannot imagine a better company for my academic and professional development. Many thanks, Joan, for teaching me how to aim for a high level of scientific integrity, for setting the bar high when it comes to ideas and execution, for sharing your personal strive for excellence, for showing me nothing is impossible when it comes to shortening the papers, for listening to my ideas and guiding my research in the best possible way. Many thanks, Emilia, for supporting my career development and my research in an invaluable way, for making me think about the impact of my work, for encouraging me to do an ISMIR tutorial and an SPM paper, for introducing fairness and bias studies in my research, for being there every time I come asking for your advice, and for showing me how to aim for being a well-rounded researcher.

I would not have been working in the field of MIR if not for Barış Bozkurt and Xavier Serra. Thanks, Barış, for your help and guidance that made me aware of this field, for pointing me in the direction of MTG, and for all your support throughout my academic journey. Thanks, Xavier, for providing me with the opportunity to join the Master's program, for all your guidance throughout my Ph.D., and for inspiring me to prioritize the motivation aspect of my research.

I have had the pleasure and the privilege of sharing an office with Juan Gómez, Helena Cuesta, and Lorenzo Porcaro. Without your great company, I surely would not look at my experience of, although limited, working in an office as joyful as I do now.

I would also like to thank Emilio Molina, for showing me a sneak-peek into life in an industrial working environment, for always supporting my ideas and developing them further, and for giving me the opportunity to have an industrial aspect in my research.

Throughout my time at MTG, I have met and enjoyed the company of countless great people. Without any particular order, I would like to thank António Ramires, Pablo Alonso, Dmitry Bogdanov, Frederic Font, Marius Miron, Perfecto Herrera, Alastair Porter, Philip Tovstogan, Minz Won, Pablo Zinemanas, Albin Correya, Xavier Favory, Andrés Ferraro, Pritish Chandna, Olga Slizovskaia, and Jordi Pons, for sharing a wealth of experiences that I will always cherish when I look back. I would also

like to thank Sonia Espí, Cristina Garrido, Seva Eremenko, Thomas Nuttall, Merlijn Blaauw, Guillem Cortès, Eduardo Fonseca, Behzad Haki, Alia Morsi, Jyoti Narang, Genis Plaja, Miguel Pérez, Pedro Ramoneda, Nazif Can Tamer, Rafael Caro Repetto, Aggelos Gkiokas, Blai Meléndez, Rong Gong, and Sergio Oramas, for leaving joyous memories with me.

I have been extremely fortunate to be a part of perhaps the greatest training network ever! Many thanks to Karim Ibrahim, Kilian Schulze-Forster, Giorgia Cantisani, Ondřej Cífka, Javier Nistal, Emir Demirel, Carlos Lordelo, Ruchit Agrawal, Alejandro Delgado, Vinod Subramanian, Luís Carvalho, and Charles Brazier, for affecting my life always in a positive way. Thanks to Alvaro Bort, Simon Dixon, Emmanouil Benetos, Geoffroy Peeters, Gaël Richard, Gerhard Widmer, Sven Ahlbäck, and Stefan Lattner, for all their efforts in making the MIP-Frontiers project possible.

During my research, I have had the privilege of working with many great researchers, from whom I have learned a great deal. Many thanks to Guillaume Doras, Christopher Tralie, Rachel Bittner, Diego Silva, Sertan Şentürk, Doğac Başaran, Brian McFee, Frank Zalkow, Meinard Müller, Erling Wold, Juan José Bosch, Mohamed Sordo, and Jan Van Balen.

Last but not least, I would like to thank my family and all my friends. I have felt their support behind me every step of the way, and I surely could not have reached this point without them.

# Abstract

One of the key practices that enrich the world's musical heritage is creating versions of existing musical works (e.g., cover songs, acoustic versions, and live performances). In addition to establishing connections between musicians, such versions provide listeners with an opportunity to rediscover a known tune. Due to a wide range of musical characteristics that may show variations between versions (e.g., key, tempo, structure, etc.), building computational systems to automatically identify versions of the same musical work is a challenging task. However, such systems open doors to many applications ranging from musical plagiarism detection in media platforms to assisting musicians in their creative process.

This dissertation aims at developing audio-based musical version identification (VI) systems for industry-scale corpora. To employ such systems in industrial use cases, they must demonstrate high performance on large-scale corpora while not favoring certain musicians or tracks above others. Therefore, the three main aspects we address in this dissertation are accuracy, scalability, and algorithmic bias of VI systems.

We first perform a large-scale analysis on the frequency and extent of the common musical changes between versions, by which we determine the crucial components of our first system. Using the insights from the analysis, we propose a deep learning–based model that incorporates domain knowledge in its network architecture and training strategy. We design explicit modules to handle common transformations between musical versions. We then take two main directions to further improve our model. Firstly, we experiment with data-driven fusion methods to combine information from models that process harmonic and melodic information, which greatly enhances identification accuracy. Secondly, we investigate embedding distillation techniques to reduce the size of the embeddings produced by our model, which reduces the requirements for data storage and, more importantly, retrieval time. After exploring potential improvements in accuracy and scalability, we analyze the algorithmic biases of our systems and point out the impact such systems may have on various stakeholders in the music ecosystem (e.g., musicians, composers) when used in an industrial context. We conclude our research by analyzing the performance of our proposed systems on two industrial use cases, in collaboration with a broadcast monitoring company.

Overall, our work addresses the research challenges of the next generation of VI systems. We show the feasibility of developing systems that are both accurate and scalable at the same time by carefully combining domain knowledge into data-driven workflows. We believe that our contributions will accelerate the integration of VI systems into industrial scenarios, and, thus, the impact of VI research on musicians and listeners will be more eminent than ever.

# Resumen

Una de las prácticas clave que enriquecen el legado musical mundial es la creación de versiones de obras musicales existentes (e.g. covers, versiones acústicas, e interpretaciones en vivo). Además de establecer conexiones entre músicos, estas versiones ofrecen a los oyentes la oportunidad de redescubrir un tema conocido. Debido a la gran variedad de características musicales que pueden verse modificadas en las versiones (ej. tonalidad, tempo, instrumentación o estructura), construir un algoritmo capaz de identificar automáticamente versiones de una misma obra musical es un gran reto. Sin embargo, tales sistemas abren las puertas a multitud de aplicaciones, desde la detección de plagio musical en plataformas multimedia, a asistir a músicos durante su proceso creativo.

Esta tesis doctoral tiene como objetivo desarrollar sistemas de identificación de versiones musicales basados en audio y aplicables en un entorno industrial. Para emplear estos sistemas en un entorno industrial, estos sistemas deben demostrar tener un alto desempeño con grandes conjuntos de datos, y a la vez no deben favorecer ciertos músicos o temas musicales sobre otros. Por lo tanto, los tres aspectos que se abordan en esta tesis doctoral son el desempeño, escalabilidad, y los sesgos algorítmicos en los sistemas de identificación de versiones.

En primer lugar ésta tesis contribuye con un análisis a gran escala de la frecuencia y la magnitud de los cambios musicales entre versiones, para lo cual se determinan los componentes clave del primer sistema resultado de ésta investigación. A partir de las observaciones de este primer análisis, se propone un modelo basado en aprendizaje profundo que incorpora conocimiento musical en su arquitectura de red y su estrategia de entrenamiento. Para ello diseñamos módulos específicos para tratar con transformaciones comunes entre versiones musicales. A continuación, escogemos dos direcciones principales para mejorar nuestro modelo. En primer lugar, se experimenta con métodos de fusión dirigidos por datos para combinar la información de los modelos que procesan información melódica y armónica, lo cual produce un importante incremento en el desempeño de la identificación. En segundo lugar, se investigan técnicas para la destilación de embeddings, con el objetivo de reducir el tamaño del embedding producido por nuestro modelo, que reduce los requerimientos de almacenamiento de datos, y lo que es más importante, del tiempo de búsqueda. Tras explorar las mejoras potenciales en desempeño y escalabilidad, se analizan los sesgos algorítmicos de nuestro sistema, y se señala el impacto que tales sistemas podrían tener en los diferentes agentes del ecosistema musical (ej. músicos, compositores), cuando se utiliza en un contexto industrial. Concluímos nuestra investigación analizando el rendimiento de los sistemas propuestos en dos casos de uso industriales, en colaboración con una compañía de monitorización de radiodifusión.

En general, la presente tesis doctoral aborda los retos de investigación de la siguiente generación de sistemas de identificación de versiones. Demostramos que es factible desarrollar sistemas que tanto acierten, como sean escalables, combinando cuidadosamente conocimiento musical en flujos de trabajo dirigidos por datos. Creemos que estas contribuciones acelerarán la integración de sistemas de identificación de versiones en escenarios industriales, y por lo tanto, el impacto de la investigación en identificación de versiones sobre músicos y oyentes será más relevante que nunca.

(*Translated from English by Emilio Molina*)

# Contents

# List of Figures

# List of Tables

# List of Abbreviations and Symbols

## Abbreviations

| Abbreviation | Description |
|---|---|
| ACR | Automatic chord recognition |
| ANN | Approximate nearest neighbor |
| AP | Average precision |
| API | Application programming interface |
| ASCAP | American Society of Composers, Authors and Publishers |
| BY-NC | Attribution-NonCommercial |
| BY-NC-SA | Attribution-NonCommercial-ShareAlike |
| CENS | Chroma energy normalized statistics |
| CQT | Constant-Q transform |
| CRP | Cross-recurrence plot |
| DAP | Detected annotations percentage |
| DLP | Detected length percentage |
| DM | Distance matching |
| DP | Dynamic programming |
| DRM | Digital rights management |
| DTW | Dynamic time warping |
| FLOPS | Floating point operations per second |
| FP | False positive |
| GRP | Gaussian random projection |
| HPCP | Harmonic pitch class profile |
| ICA | Independent component analysis |
| IFPI | International Federation of the Phonographic Industry |
| ISRC | International standard recording code |
| ISWC | International standard musical work code |
| LDA | Linear discriminant analysis |
| LSH | Locality-sensitive hashing |
| MAP | Mean average precision |

| **Abbreviation** | **Description** |
|---|---|
| MFCC | Mel-frequency cepstral coefficients |
| MIDI | Musical instrument digital interface |
| MIR | Music information retrieval |
| MIREX | Music information retrieval evaluation exchange |
| MLP | Multilayer perceptron |
| MR1 | Mean rank of the first relevant item |
| MRR | Mean reciprocal rank |
| NCA | Neighborhood component analysis |
| NP | Non-deterministic polynomial-time |
| NRP@$K$ | The percentage of the queries for which no relevant items are retrieved among the first $K$ results |
| PCA | Principal component analysis |
| PCP | Pitch class profile |
| PD | Persistance diagram |
| Prec | Precision |
| PReLU | Parametric rectified linear unit |
| RankAcc | Rank accuracy |
| ReLU | Rectified linear unit |
| SACEM | Société des auteurs, compositeurs et éditeurs de musique (Society of Authors, Composers and Publishers of Music) |
| SGAE | Sociedad General de Autores y Editores (General Society of Authors and Publishers) |
| SGD | Stochastic gradient descent |
| SHS | SecondHandSongs |
| SLI | Setlist identification |
| TP | True positive |
| TPP@$K$ | The percentage of the queries for which a relevant item (a true positive) is returned at the rank $K$ |
| VI | Version identification |

# Mathematical Symbols

## General

| Example | Symbol type | Description |
|---|---|---|
| $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}$ | Uppercase bold calligraphy letters | Tensors |
| $\mathcal{A}, \mathcal{B}, \mathcal{C}$ | Uppercase calligraphy letters | Loss functions |
| $\mathsf{A}, \mathsf{B}, \mathsf{C}$ | Uppercase sans serif letters | Sets |
| $\mathbf{A}, \mathbf{B}, \mathbf{C}$ | Uppercase bold letters | Matrices |
| $A, B, C$ | Uppercase italic letters | Constants, functions |
| $\mathbf{a}, \mathbf{b}, \mathbf{c}$ | Lowercase bold letters | Vectors |

## Specific

| Symbol | Description |
|---|---|
| $B(t)$ | Function of beat onset estimates |
| $B'(t)$ | Function of beat onset estimates $[B(t)]$ convolved with Gaussian derivative function $[G(t)]$ |
| $b$ | Pruning iterations |
| $\mathsf{C}$ | Set of all classes in the training set |
| $C^{\text{feat}n}$ | MICE model trained using "feature $n$" |
| $\mathbf{c}_i$ | Centroid for the embeddings of the class of item $i$ |
| $c^*$ | Correct class for the input item |
| $\mathsf{D}_{\text{BM}}$ | Subset of the BMAT corpora used in the large-scale retrieval experiment |
| $\mathsf{D}_{\text{DT}}$ | Subset of the Da-TACOS benchmark set used in the large-scale retrieval experiment |
| $d$ | Hyperparameter for embedding size |
| $E$ | Normalized squared Euclidean distance function |
| $E_{i,j}^{\text{DA}}$ | Distance between items $i$ and $j$ after distance averaging scheme |
| $F$ | MOVE model |
| $G(t)$ | Gaussian derivative |
| $\boldsymbol{\mathcal{H}}^{(i)}$ | The internal representation after $i$th set of convolution kernels in MOVE |
| $\check{\mathbf{H}}^{(1)}$ | The internal representation after the max-pooling layer ($M$) in MOVE |
| $H$ | Hop size |

| Symbol | Description |
|---|---|
| $\mathsf{J}$ | Set of samples in a mini-batch |
| $K$ | Cut-off rank |
| $\mathcal{L}_i^{\text{DB}}$ | Cluster matching loss using Davies-Bouldin index for item $i$ |
| $\mathcal{L}_i^{\text{DM}}$ | Distance matching loss function for item $i$ |
| $\mathcal{L}_i^{\text{G}}$ | Group loss function for item $i$ |
| $\mathcal{L}_i^{\text{N}}$ | NormalizedSoftmax loss function for item $i$ |
| $\mathcal{L}_i^{\text{NLL}}$ | Negative log-likelihood loss function for item $i$ |
| $\mathcal{L}_i^{\text{P}}$ | ProxyNCA loss function for item $i$ |
| $\mathcal{L}_{\text{A, P, N}}^{\text{Triplet}}$ | Triplet loss function for the anchor A, the positive item P, and the negative item N |
| $\mathsf{L}^{(i)}$ | $i$th set of convolution kernels in MOVE |
| $\mathbf{L}_n^{(i)}$ | $n$th kernel in $i$th set of convolution kernels in MOVE |
| $L$ | Late fusion model for the data-driven fusion scheme |
| $M$ | Max-pooling operation in MOVE |
| $m$ | Margin hyperparameter for the triplet loss |
| $n_{\text{nodes}}$ | Number of nodes to search when querying an index with the approximate nearest neighbor algorithm |
| $n_{\text{trees}}$ | Number of trees to construct an index for the approximate nearest neighbor algorithm |
| $p$ | $p$-value |
| $q_i^{(c)}$ | Logit for the class $c$ for item $i$ |
| $R(t)$ | Function of tempo-normalized local tempo deviation |
| $\mathcal{S}_{ij}^{\text{2-S}}$ | Similarity score between items $i$ and $j$ using the two-step system |
| $\mathcal{S}_{ij}^{\text{A}}$ | Refined similarity score between items $i$ and $j$ |
| $\mathcal{S}_{ij}^{\text{R}}$ | Similarity score between items $i$ and $j$ using Re-MOVE |
| $T$ | Number of frames |
| $T'$ | Number of time frames before the temporal aggregation module in MICE |
| $t$ | Time point |
| $\mathbf{v}_i$ | Embedding vector for item $i$ |
| $\mathbf{W}$ | Weights of the final linear layer of MOVE |
| $W$ | Window size |
| $\mathbf{X}$ | Feature matrix |
| $\hat{\mathbf{X}}$ | Feature matrix containing all possible transpositions of $\mathbf{X}$ |
| $\mathbf{X}_i^{\text{feat}n}$ | Feature matrix of item $i$ using “feature $n$” |

| Symbol | Description |
|---|---|
| $\mathbf{y}$ | Proxy vector for the class of item *i* |
| $\mathsf{Z}$ | Set of proxy vectors for all the classes |
| $\mathsf{Z}_i$ | Set of proxy vectors for all the classes different than the one of item *i* |
| $\alpha$ | Learnable temperature parameter in the auto-pool function |
| $\zeta_i$ | Average intra-class distance for the class of item *i* |
| $\sigma$ | Softmax function |
| $\tau$ | Temperature parameter |
| $\psi$ | Reciprocal rank metric |
| $\bar{\psi}$ | Mean reciprocal rank metric |

# Chapter 1
# Introduction

## 1.1 Motivation

"*Songs need new voices to sing them in places they've never been sung in order to stay alive.*" – Emmylou Harris, as cited by Plasketes (2005)

Creating versions of existing musical works is and has always been an essential part of musical practice. As Emmylou Harris puts it delicately, it allows songs and musical works to remain "alive" for many decades or even centuries. In fact, before the advent of recorded music, listening to a piece of music mostly meant listening to a version of it, in many cases, performed by musicians other than the original composer or performer. Versions that are reproduced faithfully with respect to the original works are typically seen as tributes to honor the composers, and versions that are altered with the limitless creativity of humans often demonstrate how an existing idea can be transformed into something that goes beyond the original intention. Regardless of the ways in which musical versions are created, they are fundamental to the world's musical heritage.

By definition, versions incorporate differences with respect to the "originals" — sometimes just some slight changes in the mix, but some other times a reimagining of all of the original components. Regardless of the degree of difference, humans have an innate ability to connect such versions to existing compositions (assuming that they are familiar) rather than considering them as independent entities (see references in Serrà, 2011). In fact, this ability is the reason why songs can stay alive through their versions.

The reason why our brains are good at forming connections between a known track and a newly heard version lies in a skill that is crucial for our cognition: categorization (Zbikowski, 2002). Humans, like many other animals, are extremely adept in this essential skill (Tversky & Hemenway, 1984). We form internal categories to understand the world around us. Without them, our brains would have been overwhelmed by the amount of information that they receive. How we create such categories is still an open question, but Rosch (1999) argues that a notion of similarity and an ability to compare objects are required. As a result of such comparisons, we can automatically form judgments regarding if an animal is a dog or if a song is a version of another.

Although categories are essential, they do not have to be unique or mutually exclusive. Different taxonomic categories can be formed based on different notions of similarity or different goals (Rosch et al., 1976; Tversky & Hemenway, 1984; Barsalou, 1991). While we can categorize living organisms based on their size, color, family, or kingdom, we can categorize pieces of music by their genres, artists, or instruments. The particular notion of similarity that we use to connect versions to each other suggests a notion of identity. Such versions may show drastic differences; yet, we somehow recognize that they share some sort of identity or essence (usually, but not always, transmitted through melody, harmony, or lyrics). As a result, we form various categories based on musical works that are familiar to us.

The ability to form categories of musical works may be a by-product of our innate categorization skills, but it is undoubtedly a valuable one. We can examine its value from three perspectives. Firstly, versions provide a rewarding experience to both listeners and musicians. For humans, connecting a new version to an existing category allows an experience of "re-hearing" a known piece of music for the first time. Such an experience positively stimulates us from two different perspectives: we hear something both familiar and novel at the same time. Our brains form expectations when hearing familiar melodies or lyrics, and versions give us a certain amount of pleasure by (although partly) fulfilling such expectations (Barrett et al., 2010; Huron, 2006; Meyer, 1956). On the other hand, the deviations from our expectations often leave us pleasantly surprised (Huron, 2006).

Like experiencing it, creating such a listening experience is also pleasing, from the artists' perspective. Many musicians, both amateur and professional, record and share their interpretations of existing tunes. Enthusiasts in online communities[1,2,3] annotate such links between versions of their favorite tracks to add value to the listening experience of others. Streaming services, in an effort to engage their users, curate special playlists dedicated to versions (e.g., "acoustic versions,"[4] "versions of Bob Dylan tracks,"[5] etc.). Taking these points into account, it is reasonable to argue that the experience of versions brings undeniable value to musical practice.

Secondly, understanding the notion of similarity that allows us to form categories of musical works is important for artists and composers to get the credit and recognition they are due. Copyright laws are designed to protect the musical works of artists and

[1]https://secondhandsongs.com/ (All the URLs shared throughout this dissertation were checked at the time of submission.)

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

[3]https://cover.info/

[4]https://open.spotify.com/playlist/37i9dQZF1DWXmlLSKkfdAk

[5]https://open.spotify.com/playlist/37i9dQZF1DX0q1RHoDiZBg

hinder cases of plagiarism. Note that the criteria for defining the identity of a musical work from a copyright perspective may be different than our innate cognitive criteria (see Section 1.2.3). However, having a grasp of abstract concepts like musical identity or essence clearly facilitates any discussion around copyright protection for musical works.

Lastly, understanding the links that connect the members of a musical category and the boundaries that separate such categories into distinct entities is an important line of research related to musicology, and music perception and cognition studies. Investigating if categories based on musical identity would apply to all types of music or if the properties of the links that connect the members of such categories are universal would surely enhance our understanding of versions.

In the light of the aforementioned points, forming categories based on musical identity (therefore, having an appropriate notion of similarity) is undoubtedly valuable to both listeners and musicians. While researchers from a wide range of disciplines including, but not limited to, neuroscience, musicology, and legal studies investigate the underlying processes and potential implications of having a notion of musical identity, computer scientists approach this interesting element of musical practice from another angle, by building systems that can automatically make judgments of similarity based on that notion.

Apart from its philosophical value (i.e., to figure out whether computers can "think," or arrive at the same conclusions as humans), the quest for automating such judgments of similarity is useful from several perspectives. Firstly, for the music ecosystem, it would be useful for the detection of musical plagiarism in media platforms (e.g., YouTube[6], Apple Music[7], and Spotify[8]) and for author and composer societies (e.g., SACEM[9], SGAE[10], and ASCAP[11]). Due to the rapid increase in the amount of new musical content created and uploaded to media platforms, automating plagiarism detection processes is becoming increasingly important. Moreover, such automated systems can be helpful for the organization of large catalogs of music. Due to certain complexities in the music industry, several rights or licenses for the same track may be managed by different parties (e.g., publishers, labels, etc.). Consolidating vast collections of music and linking related tracks (and their metadata) may facilitate the process of music licensing and prevent potential issues during such processes.

Secondly, to enhance the listening experience, such a notion of similarity can be useful for music discovery and recommendation scenarios. Above, we mentioned the plausible experience of listening to a new version of a known track. Facilitating this process may benefit both the listeners and artists, as it brings value to both parties.

[6]https://www.youtube.com/
[7]https://music.apple.com/
[8]https://www.spotify.com/
[9]https://www.sacem.fr/en
[10]http://www.sgae.es/en-EN/SitePages/index.aspx
[11]https://www.ascap.com/

The existence of many websites dedicated to sharing lists of versions of compositions can be seen as an indicator of user interest in finding versions in general. Moreover, automatically creating playlists that are dedicated to versions of certain tracks or the hits of a certain artist is a valuable service to the fans. Therefore, automating and optimizing such discovery and recommendation systems would likely be useful in industrial use cases.

Thirdly, automatically detecting versions of tracks or musical phrases can open up the possibility of conducting large-scale, data-driven musicological research. By modeling the notion of similarity that gives us an understanding of musical identity with computational tools, one may reach new insights on the subject. Moreover, by studying a network consisting of versions of the same track, phylogenetic trees of musical phrases or influences can be formed, through which researchers can delve deeper into the origins of certain milestones in the history of music.

Lastly, having systems that can assess such a quality of similarity may assist musicians in their creative processes. Quantifying how much their creations resemble existing tracks or musical phrases may be desirable when the goal is to create something truly unique. Also, an ideal system would help to understand how much change in musical characteristics is acceptable for still being considered as a version, which may guide artists that want to honor their influences while trying to express their own styles.

## 1.2 Key concepts

In this dissertation, we consider a musical version to be "any rendition or performance of an existing musical work." Although a simple one, this definition requires one to have an understanding of the term "musical work." Therefore, in this section, we define the concepts that are relevant to the research presented in this dissertation. We first introduce the computational task we address while explaining its connections to other fields of research. Then, we discuss the concept of versions from both computational and musicological points of view and introduce alternative ways of defining the concept of versions. Lastly, we introduce the concept of musical work and describe its origins and implications from computational, musicological, and legal perspectives. Although the discussions regarding musicological and legal perspectives around versions are not essential to address the task from a computational standpoint, we believe that in interdisciplinary fields of research like ours, having a broader perspective by understanding core concepts in related contexts may benefit one to gain unique insights on the problem addressed.

### 1.2.1 Task definition

Studies in audio-based music information retrieval (MIR) focus on extracting information from audio signals (tracks), which is then exploited to develop technologies that

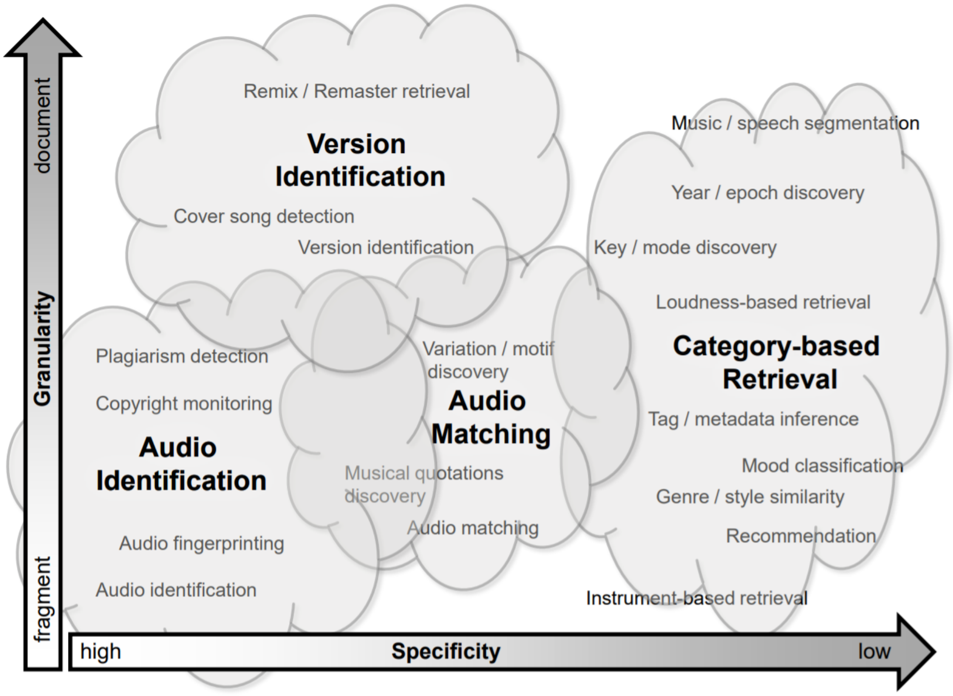


**Figure 1.1:** Specificity–granularity plane where various MIR tasks can be situated on. Taken from Grosche et al. (2012, licensed under Creative Commons BY-ND)

can be used for various applications including music retrieval, recommendation, and classification (Grosche et al., 2012). Following a query-by-example paradigm, such applications require a notion of musical similarity. However, considering the complexity of information carried by musical audio signals, defining a single notion of similarity is a rather difficult and perhaps futile goal. Therefore, the scope of musical similarity for various MIR tasks can be situated on a two-dimensional plane characterized by specificity and granularity (see Figure 1.1; Grosche et al., 2012). The two ends of the specificity axis contain high- and low-specificity systems that are differentiated by the degree of similarity between their queries and targets. While high-specificity systems aim to identify the exact musical tracks (e.g., music fingerprinting), low-specificity systems are concerned with broader descriptions of music (e.g., genre, mood, and instruments) to retrieve tracks that are related to a given query from high-level musical properties. In terms of granularity, MIR tasks are situated on a spectrum that goes from fragment- to document-level retrieval scenarios. In fragment-level scenarios, queries and targets are short fragments of audio tracks while in document-level scenarios, they are mostly entire audio tracks.

Defining the concept of similarity that connects musical versions is a challenging task. As humans, we can, in most cases, easily identify two tracks as versions of one another. However, considering the wide range of version types (Serrà et al., 2010), constructing a comprehensive similarity definition is extremely difficult[12]. Such versions may incorporate various differences in musical dimensions, including differences in timbre, tempo, structure, lyrics, recording conditions ("noise"), and so on (Serrà et al., 2010). For example, a "radio edit" of a track may have minor differences in recording quality, have sections removed, and have non-explicit lyrics, but all other musical dimensions may remain mostly unchanged. Live versions of a track often have higher degrees of variation: they may have small differences in the melody, key, and lyrics; more drastic variations in tempo, timing, structure, and timbre/instrumentation; and lots of background noise from the live recording environment. Remixes, on the other hand, may have very little in common with the original, sharing only lyrics and melody for example, which may be superimposed on musical content from a different track.

Due to such differences, the connections that link musical versions together vary depending on each case. For instance, while some version pairs may share the same melodic phrases, others may share only the lyrics. Therefore, modeling the information shared by various types of versions requires a similarity notion that encompasses multiple musical dimensions. Formulating such a notion from a computational perspective is the main focus of the line of research we refer to as version identification (VI).

Note that in this dissertation, we focus on techniques that address a wide variety of versions simultaneously. However, there are a number of subfields that are built specifically for particular types of versions. On the previously mentioned specificity–granularity plane, VI can be situated as a task that is mid-specificity and document-level, as the degree of similarity is neither based on high-level concepts nor exact characteristics of signals and the queries and targets are often entire tracks.

The first efforts toward VI emerged in the early 2000s (Foote, 2000), and it has remained an active field of research ever since. VI systems are designed in a query-by-example fashion: given a query, the goal is to retrieve all the different interpretations of the same musical composition from a corpus. The main consideration for building a VI system is to overlook the differences in musical characteristics and focus on the shared information connecting version pairs. However, instead of aiming to directly quantify this shared information, such systems create representations that are invariant to the aforementioned differences. In light of this, VI research, as other music retrieval and classification studies in MIR, benefits from the advancements from many scientific disciplines such as signal processing, machine learning, nonlinear time series analysis, computational biology, etc.

[12]We provide examples for a wide range of version types in the following webpage: https://furkanyesiler.github.io/musical_version_id_spm/ (see Appendix C)

| | Version Type | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Musical Characteristic** | Duplicate | Remaster | Radio Edit | Translation | Performance | Demo | Parody | Within-Genre | Karaoke | Live | Standard | Mashup | Acoustic | Medley | Remix | Cross-Genre | Arrangement | Quotation |
| Melody | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| Harmony | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 1 | 1 | 2 | 2 | 2 | 3 |
| Tempo | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 1 | 0 | 2 | 1 | 3 | 2 | 2 | 3 | 2 | 2 | 3 |
| Timing | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 1 | 0 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 3 | 3 |
| Structure | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 3 |
| Lyrics | 0 | 0 | 1 | 3 | 0 | 1 | 3 | 0 | 3 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 |
| Key | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| Timbre | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 |
| Noise | 0 | 1 | 1 | 1 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 2 | 3 | 3 | 3 |

| **Degree of Potential Difference** | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | Likely the same | May be variations | May be major differences | May be unrelated |

**Figure 1.2:** Examples of version types and the (subjective) degrees to which a list of musical dimensions may be altered. Both the version types and the musical dimensions are based on Serrà (2011) with a few additions. The version type "Performance" refers to a recording of a written classical work, while "Standard" refers to a recording of a folk or a jazz tune where there are often improvisational aspects.

### 1.2.2 Versions

As noted at the beginning of this section, we favor a quite simple and inclusive definition for the concept of musical versions in this dissertation. However, it is hardly the only way of defining such a concept. Therefore, we now introduce various definitions of musical versions from both MIR and musicological perspectives.

#### 1.2.2.1 Music information retrieval perspective

In MIR literature, many of the concepts around versions are overviewed by Serrà (2011). After discussing several terms like cover songs, variations, and adaptations, he argues that the term versions "globally encompasses" all renditions or reinterpretations of a musical piece, regardless of the underlying motivations, historical periods, musical characteristics, or styles. Following this idea, the dominant term that we use in this dissertation is also versions.

Our definition of the term version, which has been mentioned earlier, is chosen to be as inclusive as possible toward different styles or traditions of music. To categorize such versions, we extend the taxonomy proposed by Serrà (2011) from 10 to 18 classes, which are not mutually exclusive (see Figure 1.2). Although different taxonomies exist in musicology research, MIR-oriented taxonomies can be considered more in the vein of tags or descriptive labels.

Although there are established definitions and taxonomies related to versions in MIR literature, it is fair to say that most of those concepts serve practical needs and are born of empirical observations. Claiming that those definitions and taxonomies are superior to any of such proposed in musicology literature by any criteria is not a position we favor.

#### 1.2.2.2 Musicological perspective

Discussing the concept of versions and their function in musical practices from a musicological perspective is at the same time a challenging and rewarding task. On the one hand, it is challenging because there are disagreements between researchers on what the term should encompass, mainly due to studying different musical traditions and periods, questioning (or not questioning) the use of widely accepted terms like "cover songs," and forming taxonomies based on different criteria. On the other hand, it is rewarding because although versions form an inevitable part of the music listening experience of many people, understanding their importance and significance enhances the appreciation one may feel toward them even further.

Much of the musicological research regarding versions revolve around the concept of "cover song." Although it is perhaps the most recognizable term in daily culture compared to its alternatives, from an academic point of view, we should not overlook the ambiguities it carries (Mosser, 2008). Many researchers agree that the term signifies a new recording or performance of an existing track, mainly in pop and rock music genres, after the 1950s (Plasketes, 2005; Avram, 2011). However, other researchers argue that the track that is "covered" needs to be a "hit" track (Cooper, 2018), that cover songs have to be recorded as "tributes" to the originals (Avram, 2011), or that the concept of "covering" dates back to the 1920s (Cooper, 2018). Also, it is not clear whether the concept exists in genres like jazz and classical music or not (Gracyk, 2013). Should live performances be considered as covers? (Gracyk, 2013) Does the term apply to instrumental music? (Mosser, 2008) Apart from such ambiguities regarding the definition of the term, some researchers offer different taxonomies while categorizing types of covers (Magnus et al., 2013; Mosser, 2008). To avoid such discussions around the definition and typology of cover songs, we continue our discussion using the term "versions," to be as inclusive as possible with respect to different genres, decades, and categorizations of music.

Regardless of the ways in which we may define versions, one crucial fact remains the same: they form connections to existing pieces of music (Cusic, 2005; Plasketes, 2005; Gracyk, 2013). Creating versions is a powerful act that has many positive outcomes for the artists on both sides of the established relationship. For the artists that perform the versions, it is a means to, in most cases, pay respect and show the influences that shaped their music (Gracyk, 2013; Ortega, 2021). Through their versions, they may teach their young listeners a history lesson by pointing them to artists that came before them (Cusic, 2005). They may also contribute to a song's survival by being a "reminder," which Plasketes (2005) finds perhaps the most essential aspect of versions. Such references to existing pieces of music reflect a complex act of communication that asks the listeners to consider the new version in light of the "original," and such an act could only get embraced by the masses when the pieces of music that are being referred to could be available in recorded form (Gracyk, 2013).

Nowadays, the listeners' perception of versions is mainly positive. The artists fully embrace the act of paying respect to their influences, and their audiences respect this act. In YouTube, many musicians, both amateur and professional, record themselves playing versions, and this trend of "YouTube covers" is recognized as an entirely new genre by some critics (Constandinides, 2019). However, listeners and musicians did not always express a positive opinion toward versions (Avram, 2011; Cooper, 2018; Dineley, 2014). Mainly after the success of artists like Bob Dylan, the Beatles, and the Rolling Stones, who were writing and performing their own materials, the idea of considering musicians who perform others' tracks being lesser musicians started to become widespread (Gracyk, 2013). Although some genres like jazz and Western classical music were clearly not affected by such opinions, many genres in popular music were strictly requiring artists to write their own materials if they were to succeed (Cusic, 2005). Such a view that undervalues the talent of non-composing musicians has mostly been abandoned in recent decades.

It should be well-established by now that musical versions bring unequivocal value to both the musicians and listeners. However, excluding the business perspective in this discussion would be naive. Although the practice of "versioning" is at the core of some traditions like folk or Western classical music, it mainly emerged as a business strategy in popular music (Gracyk, 2013; Cusic, 2005; Plasketes, 2005). Especially the term "cover song" comes from the act of "covering" a hit track with other musicians. In the 1940s and 1950s, this strategy resulted in White musicians like Elvis Presley and Pat Boone recording the hit tracks of Black musicians and taking credit (Cooper, 2018). Even in cases that are not racially motivated, covering a hit track that is already proven is a shortcut to more sales for music labels (Cusic, 2005). Nevertheless, this does not mean that the act of creating versions does not benefit the musicians from a business standpoint. The artists who create the versions increase their chances of success, as they give their audiences something that they may be familiar with (Avram, 2011). On the other hand, for the artists whose tracks are being versioned, this act prolongs the life of their tracks, and it may even lead to their tracks ending up as "classics" at one point (Cusic, 2005).

### 1.2.3 Musical work

Understanding the relationship between musical versions typically indicates an intuitive understanding of the concept of musical work. We tend to extend such relationships in a transitive manner: for example, if tracks A and B are versions, and tracks B and C are versions, we typically think that tracks A and C should be versions, too. By summarizing such transitive connections, we may easily think that tracks A, B, and C are different interpretations of the same musical work. However, such an intuitive understanding may fail in certain discourses. For that reason, we now overview various ways of defining the concept of musical works in MIR, musicological, and legal contexts.

#### 1.2.3.1 Music information retrieval perspective

Like musical versions, the concept of musical works is considered from a pragmatic view in MIR research, without strict definitions that limit the use of the term. Generally, in MIR tasks that are computational in nature, the evaluation of whether a system performs a task sufficiently or not is done by using a dataset that is annotated according to the task at hand. Ideally, such annotations should be supplied by the experts, according to the proper definitions of labels. However, in lines of research where large collections of data are required, such data are mostly annotated by nonexperts (e.g., listeners, users of a platform, and enthusiasts).

Data-driven VI research (see Section 2.3.5.2) is among the MIR tasks where the annotated data is acquired from online communities, the largest one being SecondHandSongs[13] (SHS). Although the editors of the website follow certain conventions regarding the process of annotation, it may be difficult to pinpoint the basis of such conventions in the formal literature around versions. The advantage here is that, as mentioned earlier, humans have an extraordinary ability to make judgments about whether two pieces of music are versions of each other or not. With that, the quality of version annotations provided by experts and nonexperts is likely to be more similar than that of melody or chord annotations provided by those groups, as the latter concepts require a deeper understanding of musical constructs.

An important point to note about VI research is that the systems estimate similarities between a set of tracks in a pairwise manner. Although we mostly consider that the connections between versions have a transitive property and, therefore, they form sets (or cliques) of musical works, the system evaluations consider only the pairwise labels ("version" or "non-version") and ignore the concept of musical works. Nevertheless, the term musical work is often used in MIR research, although it is noteworthy to point out that the way it is defined may not have any practical significance.

#### 1.2.3.2 Musicological perspective

The MIR research lacks a formal definition of the musical work concept, and, contrarily, the musicology literature contains several. Before going into the differences in the definitions, we can note that perhaps the only thing that all researchers agree on is that the musical works are of an abstract nature: we perceive them through their

[13]https://secondhandsongs.com/

instances, be it their scores or performances; yet, their existence is independent of such instances (Goehr, 1994; Levinson, 1980).

Setting aside the nature of musical works, arguments regarding their scope is the chief issue that divides musicologists. One school of thought argues that the musical work concept only applies to Western classical music after the 1800s, and it emerged within the "romantic, aesthetic theory of fine art" (Goehr, 1994). They argue that it is a "thick" concept that must be represented in the form of a written score with its entirety, and the performers should comply with such scores perfectly.

Such a strict definition clearly does not apply to any musical genre other than Western classical music. However, another school of thought argues that the musical work concept exists in popular music and jazz (Dodd, 2014; Fisher, 2018; Butler, 2010). They argue that although the term "work" is consciously avoided when discussing popular music, the ideas and implications of it clearly exist in popular music practices (Butler, 2010). In fact, the existence of the concept of versions implies that the musical work idea is applicable in popular music because such versions are seen as entities related to the originals and not independent ones (Barron, 2006). As of Western classical music, the composers of popular music can be thought to "own" their works.

Apart from the differences in musical genres, arguing that the musical works exist only in written form may increase the significance of certain elements of music (e.g., melody, rhythm, and lyrics) while hindering the significance of others (e.g., instrumentation; Brauneis, 2014). It is natural that the written scores were the dominant form of creating an identity of musical works; however, the availability of recorded music expands the possibilities regarding what should be considered as a part of a work's identity (Brauneis, 2014).

Regardless of the discussions around its origins and connotations, the term "musical work" is an easily understood and useful one. To isolate it from any controversial opinion, we can simply follow Tagg (2000)'s description of the term, which follows as *"a musical continuum of determinate duration and of sufficient internal structural cohesion as to be understood as sonically identifiable in itself from whatever precedes or follows it, as well as from other similarly integral sets of sequences of musical sound."*

#### 1.2.3.3 Legal perspective

Even if researchers do not always agree on the definition and scope of the musical work concept, the majority of the music industry should at least recognize the importance and implications of the term from a legal perspective, which does not differentiate genres. The key concept in the legal discussions of musical work is the concept of copyright. Here, we attempt to briefly discuss the relationship between musical work and copyright concepts for two main reasons: (1) as mentioned earlier, one of the key motivations for creating versions of existing pieces of music is business-related, and

(2) detecting copyright infringements is one of the main goals of VI systems.

The concept of copyright for musical works formally emerged in 1777 in England, as a result of a legal dispute involving Johann Christian Bach, the youngest son of Johann Sebastian Bach (Carroll, 2005; Barron, 2006). The court decided that the musical scores are also subject to the copyright protection that was granted to literary works a few decades ago. This helped to establish (or to reinforce) the musical work concept, as the composers could legally own and limit the use of their materials. While increasing their wealth was certainly among the key motivations of composers, another important outcome was that they now could control the means by which their materials would be performed to the public (Carroll, 2005).

The modern copyright laws that are applied to music distinguish between two types of material: musical work and sound recording (Carroll, 2003; Day, 2011). For a long time, musical works were required to be in written form, which consequently affected their definition from a legal perspective (Brauneis, 2014). The characteristics that identify a work were melody, harmony, rhythm, and lyrics. The sound recordings, on the other hand, were instances that capture certain performances of such works (Carroll, 2003).

To apply for copyright protection, the artists had to submit both the musical work in writing and the sound recording in physical form (e.g., a phonogram). Nowadays, however, in many jurisdictions, recorded music is accepted to represent both sound recording and musical work. Therefore, submitting a transcribed version of a track is no longer needed. While this facilitates the copyrighting process for the artists and other parties, it also means that the elements of music that are protected under the musical work concept now include previously ignored elements like instrumentation and the recording process (e.g., mixing and mastering). This effectively changes what the musical work means from a legal and perhaps a musicological perspective.

## 1.3 Scope and objectives

The main goal of this dissertation is to develop systems that can automatically assess the similarity between musical tracks using a notion of musical identity (i.e., identifying whether two tracks are versions of the same musical work) for industry-scale corpora. The industrial scope of this dissertation is based on the fact that the research was carried out as a part of the New Frontiers in Music Information Processing (MIP-Frontiers) project, which is a research project funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 765068. One of the main objectives of MIP-Frontiers is to encourage collaboration between academia and industry so that the outcomes of academic research can foster industrial development and, as a result, have a larger impact.

To enable the use of audio-based VI systems in industrial use cases, such systems must demonstrate satisfactory performance on large-scale corpora while not being biased toward certain musicians or tracks. Therefore, the three main aspects we aim to address in this dissertation are accuracy, scalability, and algorithmic bias of VI systems. We hypothesize that incorporating domain knowledge–inspired design decisions into data-driven workflows can blend the strengths of both knowledge- and data-driven perspectives to result in systems that are both accurate and scalable. Since such systems would have an impact on various stakeholders in the music ecosystem (e.g., musicians and composers), it is highly important to be aware of the potential issues of such systems with respect to their inherent biases based on any given criteria (e.g., gender, popularity, etc.).

## 1.4 Summary of the contributions

This dissertation contributes to the field of audio-based music information research, with a specific focus on the areas of machine learning for musical similarity applications, music recognition, data-driven music retrieval, and fairness and bias studies in musical use cases. The main contributions of this dissertation can be summarized as follows:

C.1 A discussion of the concepts of "musical version" and "musical work" from MIR, musicological, and legal perspectives, in an attempt to bridge the gap between the semantics of those concepts in different but related contexts.

C.2 A survey of the key ideas, techniques, datasets, and evaluation methods that were proposed throughout two decades of VI research. To the best of our knowledge, it is the only review that classifies individual modules of the VI systems into an existing taxonomy of building blocks since 2010. Moreover, it is the only survey that encompasses both the knowledge- and data-driven VI strategies.

C.3 The first large-scale quantitative analysis on the frequency and extent of changes in musical characteristics between pairs of versions.

C.4 A novel data-driven VI system that incorporates explicit, musically motivated modules for handling the common changes in musical characteristics between versions. Apart from such modules, the system also uses VI-specific data augmentation strategies for improved robustness, and it demonstrated state-of-the-art performance at the time of the publication.

C.5 A data-driven fusion approach to VI for achieving accuracy improvements by combining the information from systems that process harmonic and melodic characteristics of tracks.

C.6 Investigating methods for speeding up the retrieval phase of VI systems without compromising the performance. A comparative analysis of existing and proposed embedding distillation techniques for shrinking the embeddings obtained from VI systems to reduce required computations.

C.7 An investigation into the inherent algorithmic biases of five VI systems of different characteristics (e.g., melody- vs harmony-based, knowledge- vs data-driven) using six relevant attributes of musicians or tracks (gender, popularity, country, language, year, and prevalence). To the best of our knowledge, it is the first work ever that studies algorithmic biases in a VI context.

C.8 Studies on the efficiency of our systems for industrial use cases in collaboration with BMAT, a broadcast monitoring company:

C.8.1 Identifying the musical content in long audio tracks of live music events by developing an end-to-end system that uses one of our VI models for the music recognition module.

C.8.2 A large-scale retrieval study using an industrial corpus of close to one million tracks.

C.9 A critical discussion on the current open issues and challenges of VI systems along with potential future directions.

Apart from the main contributions that are outlined above, we created and publicly shared three datasets, five code repositories, a website, and a system demo and presented a tutorial at an international conference (see Appendix C). The first dataset is curated for model development and evaluation, and it includes +110 k tracks with predefined training, validation, and test partitions, the related metadata, and a wide range of pre-extracted features. The second one is designed to investigate the algorithmic bias in VI systems and includes +37 k tracks with pre-extracted features, the related metadata, and annotations regarding gender, popularity, country, language, year, and prevalence. The third one is designed for the setlist identification task and includes metadata and timestamp annotations for 75 concerts in total and their respective reference tracks. The software repositories include the necessary code and model weights to reproduce the experiments explained in this dissertation. The website includes examples and other information regarding musical versions and aims to facilitate understanding some of the concepts mentioned throughout this dissertation.

## 1.5 Dissertation outline

This dissertation contains nine chapters each of which, excluding the introduction and the conclusion, is based on our publications. We now briefly describe the structure of the following chapters.

Chapter 2 (based on Yesiler et al., 2021a) includes a survey of the VI research in the last 20 years. We start by giving an overview of the main trends organized as periods of five-year intervals (Section 2.2). We then introduce the main building blocks of the VI systems and describe various techniques used for each of the five components (Section 2.3). We continue with explaining ideas that are independent of the building blocks and were explored for achieving improvements in accuracy and scalability (Section 2.4). We conclude this chapter with an overview of the publicly available datasets and evaluation metrics used in VI research (Section 2.5).

Chapter 3 (based on Yesiler et al., 2019) starts with a brief overview of the VI research when we first started to work on this dissertation. Based on our observations, we point out four issues to address before developing our VI systems. Firstly, we curate a dataset with more than 110 k tracks for training and evaluating data-driven VI models (Section 3.2). Secondly, we perform a large-scale quantitative analysis on the frequency and extent of observed changes in musical characteristics, which helps us to decide on which variations to address while developing our systems (Section 3.3). Thirdly, we implement several state-of-the-art systems and conduct an initial benchmarking study on our newly curated evaluation dataset to set baselines for our experiments (Section 3.4). Lastly, we analyze the performance gains obtained from using data-driven input representations rather than hand-crafted ones (Section 3.5).

Chapter 4 (based on Yesiler et al., 2020a) describes our first data-driven VI system developed for this dissertation. Based on the findings from the previous chapter, we explore incorporating musically motivated design decisions into a deep learning–based workflow. The resulting model, called MOVE, encodes each track into a fixed-size embedding vector regardless of the track duration. With this, similarity estimation between tracks reduces down to a simple Euclidean distance computation, which allows such a system to be used in large-scale retrieval scenarios. We explain our network architecture, which contains explicit modules for achieving transposition and structure invariance; and training strategy, which is based on a contrastive learning paradigm and includes data augmentation strategies specifically designed for VI (Section 4.2). We then evaluate our model by first conducting an ablation study to justify our design decisions, and then by comparing it to the state-of-the-art VI systems (Section 4.3).

Chapter 5 (based on Doras et al., 2020) consists of our investigations toward further improving the accuracy of MOVE by building a data-driven fusion system that exploits multiple sources of information. Although MOVE outperforms many state-of-the-art VI systems, one of its main drawbacks is that it processes only harmonic information from musical audio signals. Considering the complexity of variations between versions, relying on only harmonic content is clearly suboptimal. Therefore, we develop a VI system that can use any input representation regardless of its shape and explore ideas of combining systems that process melodic and harmonic information (Section 5.2). We create baselines by evaluating the systems that process single input features and compare the performances of an ensemble system and a data-driven fusion system with respect to those baselines (Section 5.3).

Chapter 6 (based on Yesiler et al., 2020b) contains a comparative study of techniques to further improve MOVE from a scalability perspective. For this, we assess a variety of embedding distillation techniques, both existing and proposed ones, to reduce the size of the embedding vectors our model encodes each track into, without compromising identification performance. For this, we first introduce a range of embedding distillation techniques, two of which are proposed by us (Section 6.2). We then evaluate their performances and choose the one that yields the best performance as a more scalable alternative to MOVE (Section 6.3). Lastly, we compare this newly selected model, called Re-MOVE, with the state of the art.

Chapter 7 (based on Yesiler et al., 2022) presents our efforts toward creating a framework for evaluating the algorithmic bias of VI systems. In the potential case where such systems are used in industrial applications, we expect them to surely have an impact on certain stakeholders in the music ecosystem. By acknowledging the fact that they are socio-technical systems rather than isolated technologies, we formulate an analysis framework where we assess whether our systems favor certain groups of musicians. We choose five VI systems of different characteristics and six attributes for our analysis: gender, popularity, country, language, year, and prevalence (Section 7.2). After presenting the results of a large set of experiments, we discuss our hypotheses on the possible reasons for the observed disparities and a possible interpretation of such results from a hypothetical fairness scenario (Section 7.3).

Chapter 8 details our experiments for applying one of our models (Re-MOVE, in particular) on two industrial use cases. Firstly (based on Yesiler et al., 2021b), we take on the setlist identification task, where the goal is to identify the musical content, ideally along with their start and end timestamps, in long recordings of live music events (Section 8.2). We develop an end-to-end workflow that includes a VI system to identify the content in overlapping windows of queries (Section 8.2.2.3). We also develop a false positive filtering algorithm that combines heuristic- and learning-based methods (Section 8.2.2.4). We compare the performances of three VI systems in a wide range of use cases with varying audio qualities and genres (Section 8.2.3). Secondly, we perform a large-scale retrieval study using a reference corpus of more than 850 k tracks (Section 8.3). After comparing two VI systems in terms of performance and runtime, we explore potential gains in accuracy by using both systems sequentially one after the other (Section 8.3.3.1). Lastly, we experiment with an approximate nearest neighbor search algorithm to further improve the scalability of Re-MOVE (Section 8.3.3.2).

Chapter 9 concludes this dissertation. We begin by summarizing the main objectives and ideas that we aimed to pursue in this research and discuss our progress toward them (Section 9.1). Then, we go over our main contributions and highlight the key results for each (Section 9.2). Finally, we provide a discussion on the current open issues, challenges, and future directions for VI research (Section 9.3).

Chapter 2

# Scientific Background

## 2.1 Introduction

In this chapter, we provide a review of the key ideas and approaches proposed in 20 years of scientific literature around VI research. For more than a decade, VI systems suffered from the accuracy–scalability trade-off, with attempts to increase accuracy that typically resulted in cumbersome, non-scalable systems. Recent years, however, have witnessed the rise of deep learning–based approaches that take a step toward bridging the accuracy–scalability gap, yielding systems that can realistically be deployed in industrial applications. Although this trend positively influences the number of researchers and institutions working on VI, it may also result in obscuring the literature before the deep learning era. To appreciate two decades of novel ideas in VI research and to facilitate building better systems, we now review some of the successful concepts and applications proposed in the literature and study their evolution throughout the years. To facilitate understanding some of the concepts mentioned in this chapter, we include audio examples on a supplementary website[14] (see Appendix C).

This chapter is based on Yesiler et al. (2021a)[15] and organized as follows. We give a chronological survey of the evolution of VI systems in Section 2.2. We introduce the building blocks of VI systems and describe them in detail in Section 2.3. We then introduce a set of ideas that can be used in any VI system regardless of their building blocks in Section 2.4. Finally, we overview the publicly available datasets and evaluation metrics for VI in Section 2.5.

[14] https://furkanyesiler.github.io/musical_version_id_spm/

[15] © 2021 IEEE. Reprinted, with permission, from Yesiler, F., Doras, G., Bittner, R. M., Tralie, C. J., & Serrà, J. (2021). Audio-based musical version identification: Elements and challenges. *IEEE Signal Processing Magazine*, 38(6), 115–136.

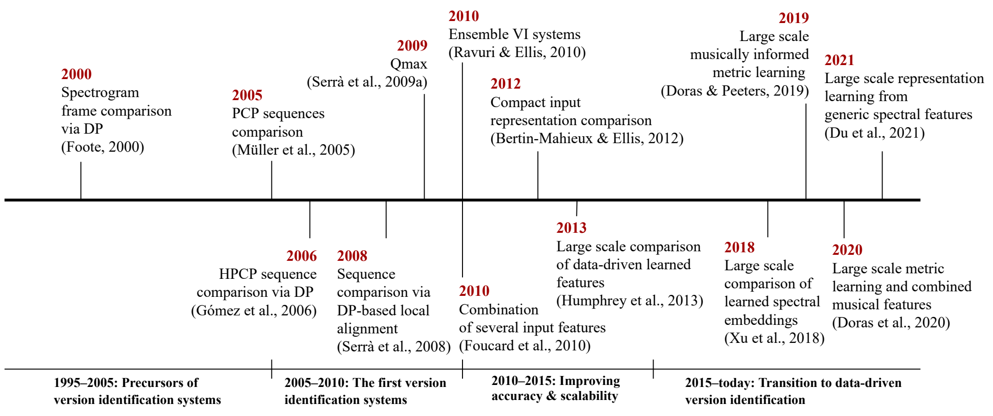

2000
Spectrogram frame comparison via DP
(Foote, 2000)
2005
PCP sequences comparison
(Müller et al., 2005)
2009
Qmax
(Serrà et al., 2009a)
2010
Ensemble VI systems
(Ravuri & Ellis, 2010)
2012
Compact input representation comparison
(Bertin-Mahieux & Ellis, 2012)
2019
Large scale musically informed metric learning
(Doras & Peeters, 2019)
2021
Large scale representation learning from generic spectral features
(Du et al., 2021)
2006
HPCP sequence comparison via DP
(Gómez et al., 2006)
2008
Sequence comparison via DP-based local alignment
(Serrà et al., 2008)
2010
Combination of several input features
(Foucard et al., 2010)
2013
Large scale comparison of data-driven learned features
(Humphrey et al., 2013)
2018
Large scale comparison of learned spectral embeddings
(Xu et al., 2018)
2020
Large scale metric learning and combined musical features
(Doras et al., 2020)
1995–2005: Precursors of version identification systems
2005–2010: The first version identification systems
2010–2015: Improving accuracy & scalability
2015–today: Transition to data-driven version identification

**Figure 2.1:** Milestones of VI research over the past 20 years.

## 2.2 A historical survey of version identification systems

This section aims to provide a survey of the evolution of VI systems across 20 years of research (see Figure 2.1 for a timeline overview of important milestones). We discuss how pioneering VI approaches were inspired by earlier music retrieval systems and how they were continuously improved over time to address the complex VI use case. Throughout this section, various system components will be referenced which are later explained in Sections 2.3 and 2.4.

### 2.2.1 1995–2005: Precursors of version identification systems

Depending on the context, musical similarity is typically assessed from editorial or social metadata such as tags or genre, from symbolic data such as Musical Instrument Digital Interface (MIDI)[16] representations, or from the audio content itself such as waveform or spectral representations. Considering such kinds of similarity assessments, pioneering works of VI mainly relied on symbolic and audio data.

**Comparing discrete sequences —** In the mid-1990s, pioneering music retrieval systems originally relied on symbolic musical representations, for instance, MIDI. In this formalism, a musical excerpt was described as a series of symbols: for instance, a monophonic melody could be described as a sequence of $n$-grams of intervals between consecutive notes, and the same principle was extended to polyphonic lines, encoding relative pitch values and durations in the $n$-gram. The similarity between musical excerpts, each represented as a series of symbols, was then evaluated with standard text-based comparison methods, such as regular expressions. However, symbolic representations only exist for very specific corpora, while audio content is now commonly available and often the main source of musical creation.

[16]https://en.wikipedia.org/wiki/MIDI

The first attempts to establish musical similarity directly from audio content aimed at reducing the problem to the already known symbolic case. For instance, for query-by-humming applications, a short hummed or whistled audio input was processed with a pitch tracker, and intervals between consecutive pitches were encoded into a series of symbols and used to query a corpus of musical scores encoded in the same way. The more complex case of polyphonic audio content was also reduced to the monophonic symbolic case by extracting a sequence of the most salient pitches.

The conversion of salient pitches into sequences of symbols ultimately relied on the limited efficiency of the then-available pitch estimation algorithms. It quickly appeared that they were not accurate enough for strict string matching, which motivated the introduction of sequence comparison algorithms based on dynamic programming (DP; see Section 2.3.3). The principle was to estimate the similarity between two sequences by counting the symbol insertions or deletions that are required to align them. This method was well-suited for musical sequence comparison because symbol (e.g., note) insertion and deletion could be penalized based on musical plausibility (e.g., subsequent notes could be considered more likely to be within a small interval; thus, large intervals could be more heavily penalized). Text-based comparison methods were abandoned in favor of DP comparisons, and these became the *de facto* standard for musical sequence comparison (see Doras, 2020 for a summary).

However, symbolic representations are inherently discrete and turned out not to be expressive enough to embed all the musical complexity of real audio content. This fostered the development of alternative, real-valued representations.

**Comparing real-valued sequences —** One of the first attempts at using real-valued representations for VI purposes was proposed by Foote (2000). The idea was to represent an audio excerpt by its energy profile (the root-mean-square signal power over short windows) and compare the resulting real-valued sequences via DP.

In a more musically motivated approach, another idea was to model music as a stochastic process, in particular, as a Markov model where each state corresponds to a chord. An audio excerpt was then represented as a Markov chain, and its similarity with others was assessed with an appropriate metric, such as the Kullback-Leibler divergence. To estimate chords more accurately, Bello & Pickens (2005) proposed training a hidden Markov model based on a simple chord vocabulary using an initial beat-synchronous pitch class profile (PCP, also known as “chroma”) sequence (see Section 2.3.1). The idea was to use this model to infer the most probable chord sequence that could have generated an observed PCP sequence and to use such a chord sequence as a proxy to evaluate musical similarity.

It then appeared that the PCP sequence itself was particularly well-suited for musical comparison: it could be deterministically computed directly from the audio and adequately represented the relative intensity of each pitch class of the equal-tempered scale. This idea was used by Müller et al. (2005), who proposed assessing the similarity between audio excerpts by comparing PCP features in a frame-wise manner, achieving tempo invariance using several representations of the same excerpt computed at different sampling rates.

Although only tangentially related, music fingerprinting algorithms (Cano et al., 2005) were proposed around the same time, which hash information from characteristic audio "landmarks" (i.e., patterns of large spectral peaks). They focus on efficient retrieval of the music recordings that match the query in an exact or near-exact manner, but they were not designed to be invariant to tempo, timbre, or pitch changes. Thanks to their efficiency, they became the fundamental technology for the music recognition industry. However, since they focus on exact or near-exact matches, they tend to fail at music recognition cases where versions are involved.

### 2.2.2 2005–2010: The first version identification systems

In the mid-2000s, the topic of musical similarity was being studied from many dimensions: from music fingerprinting to classifying musical genres. At the same time, VI started to gain more attention from the community. Pioneering attempts were logically built upon existing music retrieval approaches: for instance, using pitch trackers to extract dominant melodies (see Section 2.3.1) as a discrete input representation was one of the first proposals for VI. However, the complexity of the task pushed researchers toward exploring solutions that were better suited for this particular problem.

**Comparing harmonic features —** Most of the first robust VI methods followed the same principle: they used a harmonic progression (typically a sequence of enhanced PCP), transposed it to ensure key invariance (see Section 2.3.2), and computed a similarity score between pairs of those resulting sequences. For instance, Gómez et al. (2006) proposed the use of an enhanced PCP-based representation, the harmonic pitch class profile (HPCP; see Section 2.3.1). Key invariance was achieved by normalizing HPCP with its estimated global key, and similarity was assessed via DP. The same year, an approach involving beat-synchronous PCP representations (see Section 2.3.3) was proposed and later elaborated by Ellis & Poliner (2007). Key invariance was achieved using each possible relative PCP rotation, and similarity was assessed via cross-correlation between transposed sequences. This method yielded the best performance on the first Music Information Retrieval Evaluation eXchange[17] (MIREX) "audio cover song identification" (i.e., VI) contest that took place in 2006. These approaches were improved further for the following 2007 and 2008 MIREX editions. For instance, Serrà et al. (2008) used another method to ensure key invariance, named optimal transposition index (OTI; see Section 2.3.2), which transposed one track relative to the other so that they share the same common global key.

**Improving dynamic programming —** In the same work, Serrà et al. (2008) also introduced DP-based local alignment with musically motivated constraints to account for tempo and structure differences between versions, obtaining state-of-the-art res-

[17] https://www.music-ir.org/mirex

ults in 2007 and 2008. The following year, Serrà et al. (2009a) adapted several concepts commonly used to study recurrences in physical or biological systems. The idea was to compare PCP sequences using a cross-recurrence plot (CRP), a representation highlighting common subsequences. The global similarity was assessed via recurrence quantification analysis measurement, which in essence quantifies the importance of the similarity patterns in the CRP. This algorithm was enhanced further using similarity transitivity: if A and B are similar and B and C are similar, then it is likely that A and C are similar too (Serrà et al., 2009b; see Section 2.4.1). The combination of this algorithm with the previous approach (Serrà et al., 2009a), dubbed Qmax*, remained the state-of-the-art method in VI for more than a decade.

### 2.2.3 2010–2015: Improving accuracy & scalability

By the early 2010s, successful VI systems based on harmonic representations and local alignment had achieved promising accuracy. However, it appeared that harmonic information was not the only musical facet that could be used to adequately model music complexity. At the same time, DP algorithms were too computationally expensive to address industrial applications and the ever-increasing size of modern music corpora.

**Improving accuracy —** A strategy for improving accuracy was to investigate alternative input features: for instance, some representations were designed to account for the characteristics of the human auditory perception system (see Doras, 2020 for a review).

Another strategy was to consider the combination of existing features: different features embed complementary information, and combining them may improve the accuracy compared with using each input feature separately (see Section 2.4.2). Several combinations were investigated: for instance, HPCP extracted from the original mixture, the separated vocals, and the separated accompaniment (Foucard et al., 2010); dominant melody, bass line, and the harmony (i.e., HPCP; Salamon et al., 2012); or timbral features and HPCP (Tralie, 2017). It was shown that these combined representations improved the accuracy compared with cases where each feature was considered alone.

A third strategy was derived from the observation that some methods performed better for certain tracks than others and that an ensemble system could blend existing systems' strengths (see Section 2.4.3). Following this line of thought, different approaches based on classifiers (Ravuri & Ellis, 2010), rank aggregation (Osmalskyj et al., 2016), and similarity network fusion (Tralie, 2017; Chen et al., 2018a) were investigated to merge scores obtained from various systems. In all cases, ensemble systems improved upon the accuracy of single systems.

**Improving scalability —** All successful VI algorithms described so far rely on variants of DP sequence comparison which scale quadratically with the length of the sequences. This time complexity quickly becomes prohibitive when querying large corpora; the sequence comparison computation with the query track must be done on the fly for every track in the corpus. The problem of scalability is common to all information retrieval systems: in order to scale, they generally require a very lightweight similarity estimation function (e.g., a simple Euclidean distance), which in turn implies a data representation that can be computed offline and conveniently stored for fast lookup (e.g., a small vector of real numbers).

A first direction to improve scalability was to transform input features into a more compact representation, ideally a lightweight matrix (or vector) that could be compared via Frobenius norm (or Euclidean distance). A popular compacting transformation was the 2D Fourier transform of the PCP sequences (see Section 2.3.2) because it provides key and time-shift invariance with respect to the original input (Bertin-Mahieux & Ellis, 2012).

Another approach to reducing the size of input features was based on the assumptions that similarity between versions mainly depends on certain parts of the audio and that comparing short segments should be more efficient than comparing an entire track (see Section 2.3.4). Different methods were investigated, for instance, detecting and isolating only the segments of interest, such as those exhibiting some degree of repetition (Silva et al., 2015).

A second direction was directly inspired by fast indexing and retrieval methods that proved their efficiency in text-based retrieval contexts: instead of comparing audio features, the idea was to devise a hashing function and to store audio hashes in an index (see Section 2.4.5). For instance, locality-sensitive hashing (LSH) was used to obtain the same hash for similar audio shingles, allowing an extremely fast lookup of musically similar audio excerpts (Casey & Slaney, 2006). Along the same vein, inverted file indexing was also adapted to the music retrieval context. The original idea was to index text documents by the keywords they contain. In a musical context, a codebook of audio-based tags plays the role of the keywords. These tags were obtained by vector clustering (Kurth & Müller, 2008) or seeding (indexing of fixed-length short regions; Martin et al., 2012).

Both lightweight input features and fast indexing yielded efficient lookup times (as fast as a few seconds on a one million track corpus) but exhibited poor accuracy compared with previous systems.

Finally, a third direction was to combine different methods in a two-step pruning approach: the first step aimed to discard tracks from the corpus that could be easily distinguished as non-versions using scalable systems (Cai et al., 2016) or "weak rejectors" (Osmalskyj et al., 2016), while the second step involved a more accurate method operating on the remaining candidates (see Section 2.4.4).

### 2.2.4 2015–today: Transition to data-driven version identification

In the mid-2010s, the VI community was confronted with a dilemma: accurate systems, which could not scale up to industry-sized corpora, versus fast systems, which struggled in accuracy and were not suitable for practical use.

Many other MIR applications during this period were also confronted with a plateau in their performance gains. Inspired by advances made in other fields (e.g., computer vision, natural language processing, and speech processing), the community initiated a paradigm shift from ad-hoc hand-crafted feature extraction toward data-driven feature learning. This new perspective created a new opportunity for VI: to build more expressive representations of the audio, while still enabling a faster similarity estimation function.

**The use of feature learning —** The representation learning paradigm aims at identifying and disentangling the underlying structures in the original data that can explain its relevant characteristics. Various attempts to learn a mid-level representation from the PCP-based descriptors were previously proposed, for instance with Markov models or *k*-means. Humphrey et al. (2013) were however the first to explicitly propose using data-driven learned features to represent a track as a single embedding vector and estimate similarities between tracks by computing Euclidean distance between such embeddings. Their method used *k*-means and linear discriminant analysis (LDA) to learn an embedding space and was the first to reach a meaningful accuracy on a very large corpus (one million tracks).

The impressive results of the data-driven learning approaches in other fields also fostered the use of representation learning in VI. A common approach became to train a convolutional neural network (ConvNet) to extract a compact representation (the embedding) out of a low- or mid-level spectral representation of the audio. For instance, Xu et al. (2018) proposed training a ConvNet mapping PCP descriptors of each version of a composition to the same class. More generic spectral representations have also been investigated, such as the constant-Q transform (CQT; Yu et al., 2019b). Different variants of ConvNets were proposed to address the tempo invariance problem, including temporal pyramid pooling (Yu et al., 2019b) or standard max-pooling that have proven their efficiency in the image domain for dealing with different image scales (see Section 2.3.3). These methods yielded promising accuracy and lookup times on a large corpus of tens of thousands of tracks.

**The rise of metric learning —** Metric learning is a subset of representation learning that aims to learn a compact data representation fitting a given similarity estimation function (e.g., Euclidean distance). The underlying motivation is that the learned representations and similarity estimation functions could yield better performances for high-dimensional data, compared with their ad-hoc counterparts. Following this idea, Doras & Peeters (2019) proposed learning an embedding of dominant melody, while Yesiler et al. (2020a, see Chapter 4) proposed an approach based on a learned

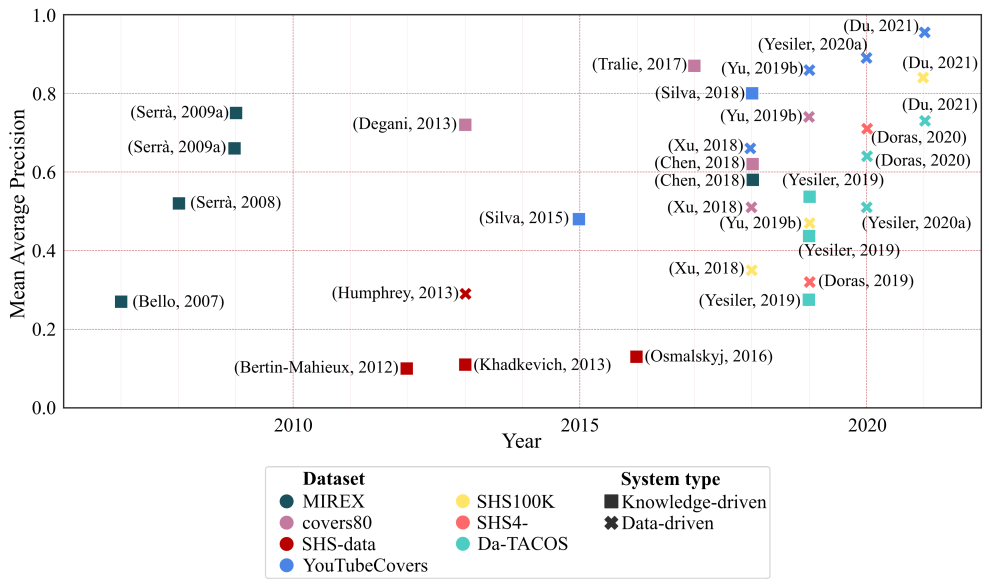


**Figure 2.2:** Performance (as measured by mean average precision) of different VI systems evaluated on several datasets (see Section 2.5.1) throughout the years.

PCP feature that approximates estimated chords. In both cases, the principle was to learn to project tracks into fixed-size embedding vectors (regardless of the duration of the tracks) so that the pairwise distance (e.g., Euclidean or cosine distance) is smaller for versions than for non-versions. This was typically achieved using an objective function such as a triplet loss, yielding promising results on datasets of up to fifty thousand tracks. In a similar vein, Zalkow & Müller (2020) proposed learning an embedding of short audio shingles and demonstrated the efficiency of this approach for Western classical music.

Recently, feature learning–based approaches have yielded the current state-of-the-art performance. On the one hand, a musically informed approach combining various complementary musical features, such as melody and harmony, yielded a competitive accuracy and lookup times (Doras et al., 2020; see Chapter 5). On the other hand, a very deep architecture applied to a generic spectral representation proved to be expressive enough to yield a similar accuracy without any prior musical knowledge (Du et al., 2021).

Throughout the last 20 years, VI systems have evolved to improve their accuracy on increasingly large corpora. This trend can be seen in Figure 2.2, which summarizes the mean average precision (MAP) scores that such systems obtained on different VI datasets over the years (see Section 2.5.1 for details about these datasets and Section 2.5.2 for details about MAP). While high performance scores could only be obtained on smaller datasets with hundreds of tracks in the early years, recent VI systems are nowadays able to reach similar performances on datasets with thousands of

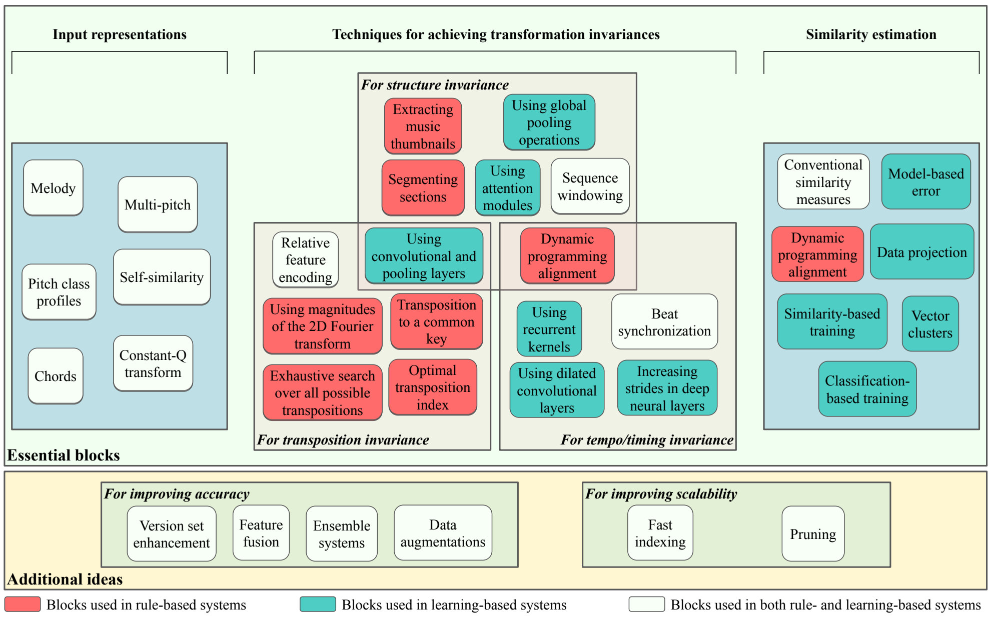

Input representations
Techniques for achieving transformation invariances
Similarity estimation
For structure invariance
Extracting music thumbnails
Using global pooling operations
Segmenting sections
Using attention modules
Sequence windowing
Melody
Multi-pitch
Pitch class profiles
Self-similarity
Chords
Constant-Q transform
Relative feature encoding
Using convolutional and pooling layers
Dynamic programming alignment
Using magnitudes of the 2D Fourier transform
Transposition to a common key
Exhaustive search over all possible transpositions
Optimal transposition index
For transposition invariance
Using recurrent kernels
Beat synchronization
Using dilated convolutional layers
Increasing strides in deep neural layers
For tempo/timing invariance
Conventional similarity measures
Model-based error
Dynamic programming alignment
Data projection
Similarity-based training
Vector clusters
Classification-based training
Essential blocks
For improving accuracy
Version set enhancement
Feature fusion
Ensemble systems
Data augmentations
For improving scalability
Fast indexing
Pruning
Additional ideas
Blocks used in rule-based systems
Blocks used in learning-based systems
Blocks used in both rule- and learning-based systems

**Figure 2.3:** Overview of the building blocks of VI systems detailed in Sections 2.3 and 2.4.

tracks.

## 2.3 Building blocks of version identification systems

Following the historical perspective presented in Section 2.2, we now present a deeper dive into VI systems by dissecting them into their building blocks. Following the literature, we consider five main components that are for (1) feature extraction, (2) transposition, (3) tempo/timing and (4) structure invariance, and (5) similarity estimation (see Figure 2.3). While each of these blocks addresses a key challenge in the VI workflow and is proven to improve system accuracy, there is no requirement that VI systems incorporate all of them. In fact, some of the techniques presented below may address multiple challenges at once.

### 2.3.1 Feature extraction

Extracting useful information from high-dimensional audio signals is the first step in VI systems. Considering the nature of the problem, the representations that are rich in relevant characteristics (e.g., harmonic or melodic) and ignore the commonly varied ones (e.g., timbre, harmonization, or noise) are favored.

**Melody —** Humans are very good at identifying a known track when the isolated melody is played. Following this intuition, melody-based representations are a natural choice for VI systems and have been explored in the literature since the early

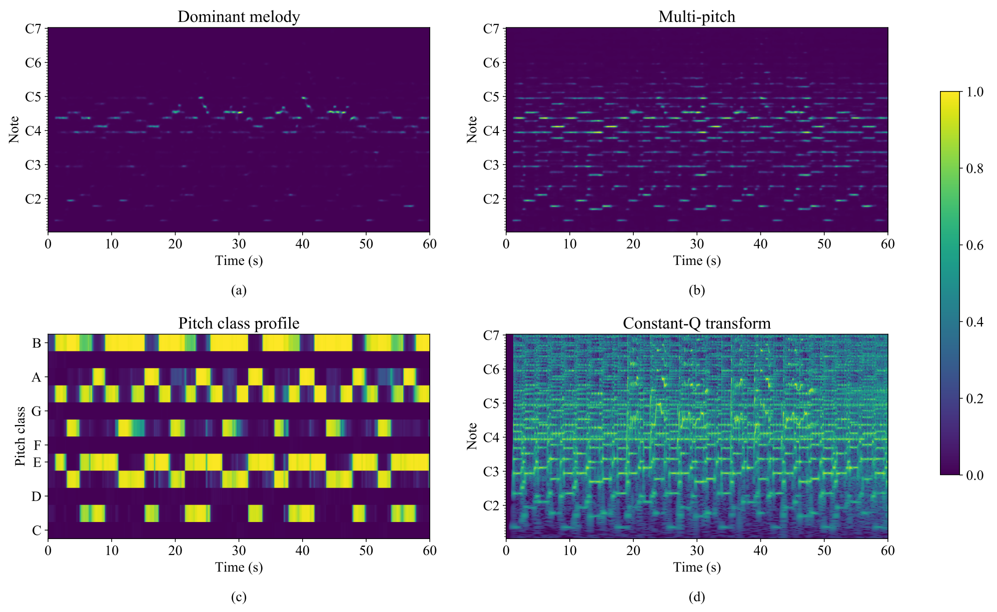


**Figure 2.4:** Common input features for VI systems, extracted for the track "Don't Stop Believin'" by Journey (included in the supplementary website). The y-axes represent musical notes (in subfigures a, b, and d) and pitch classes (in subfigure c), the x-axes represent time, and the color scale indicates the energy/intensity of such notes/pitch classes on a given time frame.

days (Marolt, 2008; Salamon et al., 2012; Doras & Peeters, 2019). Early melody estimation systems were based on subharmonic summation while the recent systems are typically implemented as ConvNets. Mainly, two types of melody representations are considered: the dominant melody, which represents a single pitch trajectory that is generated by the most dominant instrument (i.e., singing voice or a solo instrument), and the bass melody, which encodes low-frequency bass information that may be relevant for VI. An example of a dominant melody representation can be seen in Figure 2.4 (a). Melody representations are usually high-resolution features, both in time and frequency, in order to model the subtle variations generated by continuous pitch instruments like violin or singing voice. When used for VI, they are downsampled along both axes, as such levels of granularity increase computational complexity and, furthermore, incorporate detail that is detrimental to detecting variations of the same underlying musical piece.

**Multi-pitch —** Another type of high-resolution feature is a multi-pitch representation, which captures information about the pitch trajectories for each source in a track, covering both melodic and harmonic content (an example can be seen in Figure 2.4 (b)). Similar to dominant melody, multi-pitch representations are also typically downsampled along both axes and have been shown to be a useful feature for VI (Doras et al., 2020; see Chapter 5).

**Pitch class profile —** Perhaps the most exploited musical characteristic in VI systems is the harmonic content (Ellis & Poliner, 2007; Serrà et al., 2008, 2009a; Bertin-Mahieux & Ellis, 2012; Humphrey et al., 2013; Chen et al., 2018a). PCP has been the primary representation used to analyze the harmonic content in musical audio recordings for a long time. They are derived frame-wise by collapsing the energies within a certain frequency range (commonly 50 to 5,000 Hz) into an octave-independent and usually 12-bin histogram that represents the relative intensities of the 12 semitones found in the Western musical tradition (see Figure 2.4 (c) for an example). An important variant of PCP representations is the HPCP (Gómez et al., 2006; Serrà et al., 2009a), which has been, and still is, used in VI extensively. It produces a more robust summary of the tonal content than plain PCP by incorporating additional steps such as harmonic weighting and spectral whitening. Along with HPCP, another PCP variant that has been used by many systems is chroma energy distribution normalized statistics, or CENS (Müller et al., 2005). It is obtained by incorporating quantization and smoothing operations that alleviate the issues with sensitive characteristics of local PCP distributions, namely articulation variations and local tempo deviations. However, the search for more robust PCP-like features is still ongoing. A recent trend is to train neural networks to estimate "deep" PCP features from audio. For example, cremaPCP is a learned variant of PCP that estimates pitch class information needed to predict chord sequences, and it shows performance improvements over PCP and HPCP features in the VI context (Yesiler et al., 2019; Doras et al., 2020; see Chapters 3 and 5).

**Chords —** Another idea for exploiting the harmonic content is to extract chord progressions from the audio signals (Bello, 2007; Khadkevich & Omologo, 2013). They can be seen as an abstraction over PCP features to obtain a more robust summary of the harmonic content, and the fact that they can be represented as discrete codes (i.e., chord symbols) makes them highly useful for reducing computational complexity for similarity estimations and disk usage for data storage. Although the motivation for using chord progressions can be easily justified, issues in chord estimation algorithms make them less appealing for VI. For example, most research in automatic chord estimation uses a rather small target vocabulary (24 chords), which is insufficient to correctly transcribe tracks from certain genres (e.g., jazz and blues) and may lead to inaccurate input representations.

**Self-similarity —** An interesting way to make features more robust across musical versions is to consider their evolution (Tralie & Bendich, 2015; Tralie, 2017). A common way to model musical structure is via self-similarity matrices, which represent the distances between the feature vectors of each time frame and every other frame. Self-similarity matrices are attractive input representations for VI systems because they are invariant to several musical characteristics including timbre, transposition, and noise, as they only encode relative differences between features from different time frames, discarding their global offset.

**Constant-Q transform —** For many years, the input representations for VI systems were either mid- or high-level audio descriptors, mainly due to the fact that low-level representations contained too many redundant and noisy signals for developing an accurate system. However, as deep learning methods become more popular, VI researchers have begun experimenting with low-level descriptors like CQT (Yu et al., 2019b; Jiang et al., 2020). The CQT is a spectral representation of an audio signal, which is obtained by using a set of frequency filters with a constant-Q factor (see Figure 2.4 (d) for an example). The representation is quite convenient for considering pitch transpositions, as the filters are logarithmically scaled and match the pitches on the Western musical scale, which is an important advantage of CQT over other spectral representations like plain short-time Fourier transform. Considering that many deep melody- or harmony-based representations are extracted from the CQT, it has become a natural choice for deep learning–based VI systems that follow a data-driven, feature-learning paradigm.

### 2.3.2 Transposition invariance

Transposing a track to a different key is a common practice in musical performances and is among the most common variations in musical versions. Thus, it is desirable for VI systems to be completely invariant to transpositions. When not adequately accounted for, transpositions can drastically lower a VI system's performance.

**Transposition to a common key —** A straightforward idea to deal with transpositions is to estimate the key of each track and transpose them to a common key (typically C major or A minor; Gómez & Herrera, 2006). However, the accuracy of the key estimation algorithm is critical to the success of this approach, as the errors propagate to the overall system.

**Relative feature encoding —** When using dominant melody or chord representations as input, instead of using the exact frequencies or chord symbols, the same information can be represented as intervals, starting from a given offset (e.g., the first note, the mean frequency, or the first chord of a track; Sailer & Dressler, 2006; Doras & Peeters, 2019). With this, the representations are disentangled from their key offset, which results in them being transposition-invariant.

**Exhaustive search over all possible transpositions —** Another approach to this problem is to obtain similarity estimations between a track and all possible transpositions of the other track (Ellis & Poliner, 2007; Khadkevich & Omologo, 2013). Especially when using 12-bin PCP representations as input, their octave-independent characteristic limits the search space to 12 transpositions in total. This approach has the advantage of being more robust than approaches based on direct key estimation but requires a higher computation complexity in the similarity estimation step.

**Optimal transposition index —** A more computationally efficient approach for considering all possible transpositions is to estimate the OTI for a pair of tracks, which indicates the pitch transposition interval needed to make the tracks at hand the most similar (Serrà et al., 2008, 2009a). In practice, the similarity between global feature vectors of the first track and the transposed versions of the second track is estimated to find the index that results in the highest similarity. Previous works have shown that OTI is very successful with PCP features (Serrà et al., 2009a).

**Using magnitudes of the 2D Fourier transform —** The magnitude and phase components of the Fourier transform represent the energy of each sinusoidal frequency and its rotational offset, respectively. Therefore, discarding the latter provides shift-invariance with respect to the axes on which the Fourier transform is applied. In VI, applying a 2D Fourier transform on short patches of PCP features and discarding the phase is a practice that is used for obtaining representations invariant to pitch transpositions (Bertin-Mahieux & Ellis, 2012; Humphrey et al., 2013).

**Using convolutional and pooling layers —** A typical approach for achieving translation invariance in machine learning is to use convolutional and pooling layers. Convolutional layers aim to capture local information with kernels that traverse the entire input, which in the case of VI are the 2D input representations presented in Section 2.3.1. The output produced by convolutional layers can be aggregated using pooling layers so that the results are invariant to the exact location of a pattern of interest. The key point for translation invariance using this approach is to have kernel sizes much smaller than the input representations, so that the kernels can model similar patterns even when they are present at different locations of the input (Doras & Peeters, 2019). In order to take advantage of the octave-independent characteristic of PCP representations, convolution kernels can be combined with a simple pre-processing step and max-pooling operation (Xu et al., 2018; Yesiler et al., 2020a; see Chapter 4). First, PCP features are copied and concatenated in the pitch class dimension, so that the resulting representation contains all possible transpositions for each track. Next, a series of convolution operations of size 12 in the pitch class dimension are applied, generating activations for each transposition. Finally, a max-pooling layer of size $(12 \times 1)$ selects the largest value across transpositions, which can be considered as the most useful transposition, for each activation. Although proven to be useful for PCP representations, applying this technique for other input representations is not straightforward and has not been tested in the literature.

### 2.3.3 Tempo and timing invariance

Other common types of transformations in musical performances are tempo and timing changes. Tempo differences can occur across entire tracks by changing the global speed, or within certain segments with changes in local contexts. Timing differences often occur on the note level and consist of sustaining, repeating, shortening, or removing notes, mainly for conveying artistic expressions.

**Beat synchronization —** Tempo differences result from changes in the duration of bars; however, the number of bars is not affected by such changes. Therefore, by using beat-synchronous features, the temporal content of the input can be represented in units of beats, rather than units of seconds or frames (Ellis & Poliner, 2007; Bertin-Mahieux & Ellis, 2012). To compute beat-synchronous features, a beat estimation step is performed for each track, and the feature content that falls into each beat interval is aggregated to obtain one feature vector per beat. While the efficiency of this approach highly depends on the beat estimation algorithm, empirical results have proven this technique to be helpful for handling variations in tempo.

**Dynamic programming alignment —** Perhaps the most standard way of dealing with tempo and timing modifications is to perform alignment using DP algorithms. Such algorithms aim to find the optimal alignment between a pair of time series given certain constraints on the solution space. In the VI context, global alignment methods like dynamic time warping (DTW; Gómez & Herrera, 2006; Serrà et al., 2008) and local alignment methods like the Smith-Waterman algorithm (SWA; Smith & Waterman, 1981; Serrà et al., 2009a; Chen et al., 2018a) have been proven useful for achieving tempo and timing invariance. Although resulting in higher system performances compared with other strategies against tempo and timing changes, DP methods require higher (typically quadratic) computation costs.

**Increasing strides in deep neural layers —** The striding operation in neural networks is a common method for downsampling a given input, and it may be useful for being robust to timing variations (Doras & Peeters, 2019). Although prone to aliasing, this method of downsampling applied on abstract representations found in deeper layers of neural networks is commonly seen in deep learning models, as well as some VI systems.

**Using recurrent kernels —** In the machine learning community, recurrent kernels are a popular choice for dealing with sequential data. They often incorporate a notion of "memory" that facilitates differentiating between past and future events, which gives them the capacity to preserve nonstationary characteristics. The classic formulation of recurrent kernels is prone to a number of issues while training, including the inability to consider long-term dependencies and not being suitable for parallel processing. Considering that the VI systems typically process full-length tracks rather than short fragments, the issues around recurrent kernels make them less appealing for VI research; thus, they have been underexplored in the literature (Ye et al., 2019).

**Using dilated convolutional layers —** The receptive field of convolution kernels can be increased by separating kernel elements from each other by a dilation factor while keeping the number of parameters constant. Using multiple kernels with different dilation rates can help efficiently process a given input for various tempi using convolutional kernels (Jiang et al., 2020).

### 2.3.4 Structure invariance

Another common source of variance across versions is the musical structure. This includes removing, repeating, or changing the order of the existing sections, or introducing new ones.

**Segmenting sections —** An intuitive solution for dealing with structural changes is to first perform segmentation in order to identify sections and later estimate the segment-wise similarities between two tracks (Gómez et al., 2006; Cai et al., 2016). Although this is an ideal solution for achieving structure invariance, segmentation algorithms are currently error-prone, and comparing wrongly segmented sections may drastically reduce the system performance.

**Extracting music thumbnails —** To avoid the computational complexity of using all segments and performing segment-wise similarity estimation, some VI systems extract single or multiple "thumbnails," or short representative clips, for each track and use them as a representation of it (Silva et al., 2018). Such thumbnails can be selected using various criteria, the most common being selecting the most repeated subsequences. This technique assumes (1) that the most repeated section will correspond to the most informative or characteristic one, and (2) that all versions of a particular track include that section due to its importance. While these assumptions hold true for many versions of many popular tracks, extreme stylistic changes may present difficulties for this method.

**Sequence windowing —** Following the idea of comparing subsequences rather than entire tracks, this technique avoids a segmentation or thumbnailing step by dividing a representation into short, overlapping segments of a fixed size (also called shingles) using a predetermined hop length between offsets of consecutive windows (Müller et al., 2005; Casey & Slaney, 2006; Bertin-Mahieux & Ellis, 2012). After obtaining multiple shingles for each input, such shingles can be aggregated by computing their mean or median (Bertin-Mahieux & Ellis, 2012), the distances obtained between an item and multiple shingles can be aggregated (Zalkow & Müller, 2020), or each shingle can be used individually for fragment-level retrieval.

**Local alignment —** Apart from being helpful for tempo and timing invariance, alignment algorithms can also be useful for achieving structure invariance (Serrà et al., 2009a; Chen et al., 2018a). The key consideration for this is to avoid global alignment algorithms (e.g., DTW) since they struggle in cases with structural changes. Local alignment algorithms (e.g., SWA), on the other hand, are ideal candidates for achieving structure invariance due to their goal of finding alignments among subsequences.

**Using convolutional and pooling layers —** As introduced in Section 2.3.2, convolutional and pooling layers can be combined to achieve translation invariance. Such a property can be useful for achieving structure invariance as it makes the VI systems less sensitive to the exact locations of patterns (e.g., when the ordering of sections are changed). Moreover, to handle cases where certain sections of a track are repeated, using max-pooling for downsampling can be useful.

**Using global pooling operations —** It is common practice to use local pooling operations for downsampling purposes. Global pooling, however, takes the downsampling aspect a step further to aggregate information from an entire dimension to output a single value (Xu et al., 2018; Yu et al., 2019b; Doras & Peeters, 2019). The main purpose of this is to be invariant to track length and to obtain a fixed number of features per track. Choosing the pooling operation is a crucial aspect of this technique and can be done in an intuitive way. The average-pooling operation considers all the temporal frames as equally important. On the one hand, this may hurt system performance when versions include new or missing segments, but, on the other hand, the most repeated sections will contribute to the results more than the others. The max-pooling operation chooses only the time frames with the highest value for each channel, and assuming that the frames with the highest value are the most informative ones (per channel), this operation is better suited against changes in structure. However, during the backpropagation phase of training, the gradients will flow only from the selected time frames for each channel, which may introduce some instability during earlier training steps due to weights being randomly initialized and updated only through such selected time frames.

**Using attention modules —** Although pooling operations have been the most popular choice for structure invariance in deep learning–based VI, they consider each frame independently and ignore their relationships. Attention modules address this issue by inferring links between frames so that the models can select which frames to highlight or ignore based on the information contained in each frame. A popular attention technique in the machine learning community is self-attention, popularized by the Transformer architecture (Vaswani et al., 2017). Self-attention passes information about each frame to all the others and modifies the features in each time step using this information. Ideally, only the most informative frames (e.g., the most repeated ones) are highlighted, and the structural changes are overlooked. Although widely used in the applied machine learning literature, the self-attention idea has been underexplored in VI (Jiang et al., 2020). Multi-channel attention is another technique that can be considered as a link between attention and global pooling techniques. The goal is to learn a weighted average of time frames for each feature (or channels of convolutional layers) independently (Yesiler et al., 2020a; see Chapter 4). Compared with self-attention, there are two main differences: (1) different sets of kernels are trained to learn the attention weights, which are later applied to the input of the module for performing a weighted average, and (2) using convolution kernels for computing the weights provides attention only within a local context.

### 2.3.5 Similarity estimation

The main goal of VI systems is to estimate similarities between pairs of tracks in a way that versions of a musical composition return higher similarity scores than non-versions. The techniques used for achieving invariances are crucial for this purpose. However, the similarity estimation algorithm must also be chosen carefully to succeed. Based on the literature, we consider two main types of similarity estimation: knowledge- and data-driven approaches.

#### 2.3.5.1 Knowledge-driven approaches

Knowledge-driven approaches use heuristic-based algorithms that are often selected based on domain knowledge. The characteristics of the invariant representations obtained from the previous steps (e.g., whether representations lie in Euclidean space) play an important role in the decision of which algorithm to use for this final step.

**Conventional similarity measures —** When the invariant representations obtained from previous blocks are suitable, similarity measures such as cross-correlation (Ellis & Poliner, 2007), the Euclidean distance (Marolt, 2008), or the dot product (Müller et al., 2005) are simple, yet effective choices for similarity estimation.

**Dynamic programming alignment —** As described in Sections 2.3.3 and 2.3.4, DP techniques are often used for tempo, timing, and structure invariance, and they eliminate the need for adopting further similarity estimation steps as they provide a measure of it. In the cases of alignment algorithms like DTW, the cost of the optimal solution can be used as a distance measure (Foote, 2000; Gómez & Herrera, 2006; Gómez et al., 2006). In the case of local alignment algorithms like SWA, the length of the longest-aligned subsequence is typically considered as a measure of similarity (Serrà et al., 2009a; Chen et al., 2018a; Tralie, 2017).

#### 2.3.5.2 Data-driven approaches

Data-driven approaches aim to learn a function that transforms the data to facilitate similarity estimation through conventional measures (e.g., Euclidean distance). Such functions are learned using training data, in a supervised or unsupervised fashion. In supervised cases, the semantic relationships (i.e., version or non-version) between pairs of items are used for obtaining effective similarity functions. The learned functions depend on the inductive biases of the models and the training process.

**Data projection —** Along with a few classification approaches, early works for data-driven VI considered data projection algorithms like PCA and LDA (Bertin-Mahieux & Ellis, 2012; Humphrey et al., 2013). They are used for transforming invariant representations obtained in the previous blocks into more compact embedding vectors. Such systems can be considered as hybrid approaches that connect knowledge- and data-driven similarity estimation, as they incorporate rule-based algorithms for their initial steps.

**Vector clusters —** Another common approach to derive input representations was based on vector clustering, such as *k*-means. A learned dictionary of cluster centroids can be used to efficiently encode data with a small number of components. This approach has been used to encode input representations into chord series (Bello & Pickens, 2005), hashcodes (Kurth & Müller, 2008), or embedding vectors (Humphrey et al., 2013).

**Model-based error —** Another early data-driven approach to VI was to study model-based errors (Serrà et al., 2012; Foster et al., 2015). In this approach, a simple parametric model that describes the temporal evolution of the feature sequence is fit to the data. This modeling can be performed on the basis of a single musical piece or from multiple pieces that form a version clique. After that, the model is used to predict future samples of a new feature sequence coming from a candidate piece (i.e., fragments of the sequence are used as input to the model, and this outputs the most reasonable continuation). If the model produces a small error, one concludes the candidate piece is a version of the piece that was used to train the model.

**Classification-based training —** Classification-based training approaches are perhaps the most popular in supervised learning. In VI, three main formulations exist. Firstly, distance/similarity scores can be obtained from multiple systems and used to train a classifier to make a final decision (see Section 2.4.3). Secondly, a cross-similarity matrix of two tracks can be computed, and a convolutional network can be used to determine whether the inputs are versions of each other or not (i.e., binary classification; Lee et al., 2018). Although computing cross-similarity matrices introduces a computational load for the similarity estimation step, convolutional networks can replace the quadratic-complexity alignment algorithms like SWA, which results in a considerable improvement in terms of overall computational requirements. Thirdly, the training process can be formulated by considering each clique as a separate class (i.e., multiclass classification; Xu et al., 2018; Yu et al., 2019b; Jiang et al., 2020). One important consideration is that during inference, it is likely that the system will encounter tracks from cliques that are not in the training data, and using a pure classification strategy, it would not be possible to correctly identify those cases. To handle this, a typical solution is to consider the output of the penultimate layer of the model as the embedding for each track. Training then aims to make the embeddings from different classes linearly separable, which is a way of constructing a similarity function.

**Similarity-based training —** In recent years, the most popular training formulation in VI is similarity-based, using metric learning approaches (Doras & Peeters, 2019; Yesiler et al., 2020a; Jiang et al., 2020; see Chapter 4). The supervision signals in this context only require the information about whether two items are similar (as in, belong to the same clique) or not. During training, instead of predicting the classes of each item, these systems focus on manipulating the distances of items directly by pulling together similar items and pushing apart the dissimilar ones. The most popular training objectives for this approach are contrastive and triplet losses.

## 2.4 Beyond building blocks: accuracy and scalability improvements

This section introduces a set of ideas that can be incorporated in VI systems regardless of their building blocks, mainly for improving their accuracy or scalability. We group these ideas into six main categories: version set enhancement, feature fusion, ensemble systems, pruning, fast indexing, and data augmentation.

### 2.4.1 Version set enhancement

Versions of the same composition can be viewed as items of the same set, or clique. Using this intuition, community detection algorithms have been studied to refine the obtained distances between queries and items in a corpus. Specifically, one can construct a fully connected graph based on similarities obtained with any system, eliminate certain edges based on a threshold of some quantity (e.g., a distance threshold), and, assuming transitive relations, complete the missing links in this graph (Serrà et al., 2009b). This process is highly efficient and can lead to better retrieval performance by finding undetected versions and cleaning up the noisy results. As a side benefit, it is possible to use measures of centrality on these completed graphs to estimate the original performance from which subsequent versions arose (Serrà et al., 2009b).

### 2.4.2 Feature fusion

Considering the complexity of the VI task, no single feature has been able to capture all possible transformations that exist across versions. For instance, while the majority of VI systems work by matching sequences of pitch-based features, this leaves a blind spot for certain genres where the notes do not carry the dominant musical expression (e.g., 1980s hip-hop, and drum solos; Tralie, 2017). At the same time, features that ignore notes are missing crucial information that helps much of the time (Tralie & Bendich, 2015). Hence, intuitive solutions for such issues have been proposed to find ways of combining information from various musical dimensions.

**Early fusion —** Music, in general, has repeated structures that can be represented as a graph, where each node represents a small snippet of audio and edges exist with high weights between snippets that are similar, according to some chosen features. Different features can lead to different noisy observations of an ideal structure graph and, in most cases, no single feature reliably picks up on all aspects of the complex musical structure. To address this, it is possible to use a technique known as similarity network fusion (Tralie, 2017; Chen et al., 2018a) to reconstruct a cleaner graph from these noisy observations, particularly if they contain complementary information. It is possible to adapt these features so that cleaner cross-similarity measures can be obtained between versions, and this can significantly improve the system accuracy over each feature alone (Tralie, 2017).

**Late fusion —** Another possibility for feature fusion is to combine information from various features at later stages of systems. For example, cross-similarity matrices for the same pair of tracks but obtained with different features can be aggregated using simple schemes like taking the maximum or the minimum (Foucard et al., 2010). In addition, embedding vectors obtained with systems that use different features can be concatenated and projected into a new space, which can then be shaped by combined characteristics of all input features (Doras et al., 2020; see Chapter 5).

### 2.4.3 Ensemble systems

One common strategy to boost overall accuracy is to incorporate the output from multiple pipelines. Such systems, known as ensemble systems, thereby leverage the joint strength of disparate workflows. Although the motivation behind some ensemble systems is similar to that of feature fusion (combining information from various musical dimensions), here, we describe VI systems that combine multiple systems after they return distance or similarity scores between queries and a corpus of tracks.

**Training a classifier —** A first approach for aggregating scores obtained from various systems is to train a shallow classifier that takes a set of scores as input and returns a binary decision (i.e., version/non-version). Depending on the characteristics of the classifier, nonlinear relationships between the input scores may be explored. In VI, this strategy has been explored to combine systems that use different input features (e.g., PCP, dominant melody, and bassline; Salamon et al., 2012), and different similarity estimation steps (e.g., local alignment and cross-correlation; Ravuri & Ellis, 2010).

**Similarity normalization and aggregation —** In cases where the distance scores obtained from various systems are well-calibrated, aggregation of such distances can be trivial with simple schemes like taking the mean, the maximum, or the minimum. However, when there is a mismatch regarding the scale of such scores, extra operations like simple normalizations are needed to avoid the issue (Degani et al., 2013).

Another approach to handle scores of different scales is to consider the global ranking of all tracks in a corpus for each system since rankings are automatically invariant to the scale of the similarity scores obtained from a system. Rank aggregation can then be used to create a global ranking that incorporates the individual ranks from each system. The Kendall tau distance (Osmalskyj et al., 2016) is an objective function for measuring the agreement of rankings and indicates the number of pairs of ranked tracks that have a reversed order. A Kemeny optimal global ranking (Osmalskyj et al.,

2016) is a ranking that minimizes the Kendall tau distance to each individual system's rank. Unfortunately, finding such a ranking is NP-hard. However, using an initial guess based on heuristics such as mean and median rank aggregation, followed by local Kemenization, in which greedy swaps are performed until the Kendall tau distance is minimized, can lead to superior performance over individual systems in practice (Osmalskyj et al., 2016).

**Late similarity network fusion —** In addition to promoting cliques from a single system, it is also possible to fuse graphs from multiple similarity networks. Similarity network fusion can again be used to enhance cliques, but the algorithm operates at the track level instead of the time frame level (as done in feature fusion). Due to normalizations based on local neighborhoods within each network, this technique can fuse similarity measures from any set of systems, and it has been shown in practice to improve accuracy when fusing PCP-based systems that use different alignment schemes (Chen et al., 2018a), as well as between systems built on timbral and PCP-based features (Tralie, 2017).

### 2.4.4 Pruning

As many information processing systems suffer from the accuracy–scalability tradeoff, a plausible solution is to design multistep systems where the scalability and the accuracy of multiple systems may complement one another. For this, fast algorithms can prune the corpus to allow slow but better-performing systems to operate on a reduced set of data for improving the computation times.

**Scalable VI systems —** Lately, deep learning–based VI systems have made substantial contributions for bridging the accuracy–scalability gap, but before them, scalable VI systems were not sufficient for obtaining confident results. However, considering their far-from-random performances, such early systems became a natural choice to be used as the first step of pruning-based, multistep systems (Cai et al., 2016). The general tendency was to use systems that encode tracks into compact embedding vectors as the first step, mainly to take advantage of the fast lookup times. Afterward, local alignment–based systems were used on the pruned set of tracks to obtain the final results. Pruning systems nonetheless need to have good recall (at the expense of good precision, if necessary).

**Weak rejectors —** There is a multitude of features that are similar for versions of the same track, but which are not strong enough indicators to confidently label them as versions of one another. Still, having a large collection of such "weak rejectors" can be used to narrow down candidates, leading to improved scalability. Among such features are bag of words of PCP features, duration and tempo of a recording (Osmalskyj et al., 2016), and structure-related descriptors (Yesiler et al., 2019; see Chapter 3). Although not applicable in the audio-based VI literature, textual bag of words can also be applied to title and lyrics information, if they are available (Correya et al., 2018).

### 2.4.5 Fast indexing

Like every other information retrieval system, VI systems store and index tracks for future lookup and comparison. This perspective motivated other strategies to devise new music representations, inspired by text-based content indexing. Different kinds of efficient text indexing algorithms were then adapted to music retrieval: for instance, and among others, inverted file indexing and LSH.

**Inverted file indexing —** In a text-based indexing context, the idea is to establish a list of keywords (the codebook) and to use these keywords to index the documents where they appear. In a VI context, the codebook contains encodings of audio shingles, which are used as indexes to the full audio tracks. For instance, a *k*-means approach was proposed to learn a codebook from the set of all PCP vectors present in a corpus. The inverted index was built using the closest code to each PCP frame as an index to the full track (Kurth & Müller, 2008). Another proposal built the codebook encoding each PCP sequence as a major/minor chord series and then using short chord subsequences as index entries (Martin et al., 2012).

**Locality-sensitive hashing —** The basic idea of LSH indexing is to devise a hashing function that will guarantee that similar contents are encoded by the same hash with a high probability, while the probability that different contents are mapped to the same hash remains low. There are various ways to generate hashing functions that will satisfy these properties (Casey & Slaney, 2006). Several authors adapted this principle to the VI context and proposed encoding shingles of input representations with an LSH scheme [e.g., using dominant melody (Marolt, 2008), chord progression (Khadkevich & Omologo, 2013), or PCP (Casey & Slaney, 2006)].

The recent deep learning–based systems (see Section 2.3.5), which also encode tracks into compact vector embeddings, have superseded these fast indexing approaches. However, techniques like LSH could still be considered to further speed up the retrieval process of deep learning–based systems.

### 2.4.6 Data augmentation

With the increasing interest in data-driven VI systems, domain-specific strategies for robust representation learning are becoming more important. For this, we now introduce data augmentation strategies that are inspired by musical characteristics that can be modified while creating versions of a composition.

**Pitch transposition —** To simulate pitch transpositions that are typically observed between versions, different strategies can be used based on the input representation. Firstly, regardless of the input representation, a pitch shift transformation can be applied to the audio signal before feature extraction. While this operation is universal and does not depend on the type of input, it can introduce some artifacts on the signal and may require more computational resources than its alternatives. Secondly, if the PCP features are used as input representations, a simple circular shift along the frequency axis is sufficient to simulate a pitch shift operation, thanks to their octave-independent characteristics (Xu et al., 2018; Yesiler et al., 2020a; see Chapter 4). Lastly, if melody or CQT representations are used, shifting the values by a certain number of rows along the frequency axis can be useful. However, unlike PCP features, these representations are not octave-independent, and the behavior of this operation at the boundary bins (the lowest and the highest) should be considered.

**Time stretching —** As with pitch shift transformations, increasing or decreasing the tempo of a track can be done before the feature extraction step. A common alternative to this is to apply interpolation functions (e.g., linear) to the 2D input representations (e.g., PCP, melody, or CQT; Doras & Peeters, 2019).

**Frame-level alterations —** To simulate minor timing variations where some notes are sustained, repeated, shortened, or removed, similar operations can be applied to randomly selected frames from 2D input representations. For this, such frames can be duplicated, silenced (by replacing them with zero vectors), or simply removed (Yesiler et al., 2020a; see Chapter 4).

**Input patch sampling —** Similar to the idea of shingling, the input patch sampling strategy is to randomly select fixed-size patches from the input representations to use in the training process (Yesiler et al., 2020a; Yu et al., 2019b; see Chapter 4). This can be viewed as simulating structural changes where some sections are removed from a version. Note that the sizes of the patches for this strategy (e.g., 120–180 s) are generally larger than the sizes used for shingling (e.g., 30–60 s).

**Input length variation —** While applying the input patch sampling transformation, the sizes of the patches can be varied to mitigate bias toward a certain representation length (Yu et al., 2019b).

**Noise insertion —** Lastly, several transformations to audio signals can be applied to simulate the differences in recording conditions. Some examples are additive noise, low-pass filters, and MP3 transcoding.

## 2.5 Datasets and evaluation metrics

This section presents an overview of the publicly available datasets and the most widely used evaluation metrics for VI. Although there exist different datasets and evaluation methods for various subproblems within VI (see Section 9.3.5), we here focus only on the most frequently used ones.

### 2.5.1 Datasets

Finding data for developing and evaluating MIR systems is challenging, mainly due to the fact that musical audio is often subject to copyright. Historically, the impact of this issue on VI was that the researchers were limited to developing and evaluating

| Dataset | Training subset | Validation subset | Test subset | Content |
|---|---|---|---|---|
| MIREX collection | - | - | 330 (30) + 670 noise tracks | Proprietary collection |
| covers80 (Ellis, 2007) | - | - | 160 (80) | Full audio tracks and metadata |
| SecondHandSongs dataset (Bertin-Mahieux et al., 2011) | 12,960 (4,128) | - | 5,236 (1,726) | Pre-extracted features (a wide range including PCP, timbral features, beat, etc.) and metadata |
| YouTubeCovers (Silva et al., 2015) | 100 (50) | - | 250 (50) | Pre-extracted features (three PCP variants) and metadata |
| SHS-100K (Xu et al., 2018; Yu et al., 2019b) | 84,340 (4,611) | 10,883 (1,842) | 10,547 (1,692) | YouTube URLs and metadata |
| Da-TACOS (Yesiler et al., 2019) | 83,904 (14,499) | 14,000 (3,500) | 13,000 (1,000) + 2,000 noise tracks | Pre-extracted features (three PCP variants, timbral features, and four rhythm features) and metadata |
| SHS5+ & SHS4- (Doras & Peeters, 2019) | 62,311 (7,460) | - | 48,483 (19,445) | Pre-extracted features (CQT, melody, multi-pitch, and PCP variants) and metadata |

**Table 2.1:** Publicly available VI datasets. Values outside and inside the parentheses indicate the number of tracks and unique cliques, respectively.

their systems using in-house private corpora, which made unified benchmarking of systems a difficult task. These datasets had varying characteristics, such as the size of the corpus, the cardinality of cliques, the distribution of musical genres, and so on. However, with the help of online communities like SecondHandSongs (SHS), where editors and users annotate musical versions in terms of their connections with previous musical compositions, this issue is mostly alleviated today. Therefore, for the remainder of this section, we focus only on the public benchmarks and publicly available datasets that have been frequently used for VI. A summary of such datasets can be seen in Table 2.1.

**MIREX collection —** The "audio cover song identification" competition in MIREX stood out as the only platform for benchmarking in the early days of VI. The dataset used in this competition is private and includes 1,000 tracks from a variety of genres. 670 among them are considered as "noise" tracks that do not belong to the same clique as any others. The rest of the data is organized into 30 cliques with 11 versions each. While the query set consists of only those with multiple versions (330 tracks), the corpus includes the entire collection of 1,000 tracks. The inclusion of noise tracks that are not queried is done mainly to imitate the distribution of industrial corpora, and it influenced some of the forthcoming publicly available VI datasets.

**covers80 —** Apart from the MIREX collection, the first dataset curated for VI research is covers80 (Ellis, 2007), and it was released publicly as opposed to the former. It includes full-length audio files for 160 tracks divided into 80 cliques with two versions each and no noise tracks. The majority of this data is taken from the uspop2002 dataset[18], and the rest was taken from a few commercial "cover albums." The major advantage of this dataset is that it includes audio files for all tracks, which enables researchers to develop and evaluate systems that use novel input representations. However, the limited size is an important drawback as the reported results may not be statistically significant for a true comparison of systems.

**SecondHandSongs dataset —** The next publicly available dataset for VI was the SecondHandSongs dataset (SHS-data; Bertin-Mahieux et al., 2011). It is a subset of the Million Song Dataset (Bertin-Mahieux et al., 2011)[19], and the version annotations are obtained using the SHS API[20]. It includes a training set with 12,960 tracks split into 4,128 cliques and a test set with 5,236 tracks split into 726 cliques, without any noise tracks. With the release of SHS-data, VI research entered into a new era, which led to the development of scalable systems that can leverage and be evaluated on large datasets. However, due to legal issues, this dataset includes only pre-extracted features that were obtained using the, now obsolete, EchoNest API[21]. Therefore, the VI systems developed and evaluated using this dataset have a strict limitation in the input representations they can use, which may reduce accuracy and hinder system deployment in the real world due to their proprietary nature.

**YouTubeCovers —** Following the idea of sharing pre-extracted features, the YouTubeCovers dataset was released with a larger set of harmonic features compared to SHS-data (Silva et al., 2015). It includes a total of 350 tracks from 50 cliques and is further split into a training subset with 100 tracks (two per clique) and a test subset with 250 tracks (five per clique) with no noise tracks. However, having the same cliques in both training and test subsets may result in biased evaluations. Moreover, like covers80, the rather small evaluation set may lead to statistically insignificant results (Yesiler et al., 2019). Although there are still research papers using this dataset, the URL shared in the original publication for obtaining the dataset is no longer maintained.

**SHS-100K —** With deep learning–based systems getting more prominent, a need for larger datasets has emerged. Addressing this need, SHS-100K includes a total of 108,869 tracks split into 9,202 cliques (no noise tracks), which was a considerable increase compared to the largest dataset until then (Xu et al., 2018). The version annotations are collected from SHS, and the dataset includes YouTube links for the tracks,

[18]https://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html
[19]http://millionsongdataset.com/
[20]https://secondhandsongs.com/page/API
[21]https://en.wikipedia.org/wiki/The_Echo_Nest

rather than any pre-extracted features. It was initially divided into training, validation, and test subsets with 101,968 tracks (8,177 cliques), 3,918 tracks (909 cliques), and 2,983 tracks (116 cliques), respectively. However, in a later publication, the dataset was split in a different way mainly to have a larger test set, having 84,340 tracks (4,611 cliques), 10,883 tracks (1,842 cliques), and 10,547 tracks (1,692 cliques) for training, validation, and test, respectively (Yu et al., 2019b).

**Da-TACOS —** Another dataset that addressed the need for larger corpora is Da-TACOS (Yesiler et al., 2019; see Chapter 3). It includes a benchmark subset with 15,000 tracks that split into 1,000 cliques with 13 versions each and 2,000 noise tracks. Like many others, the version annotations are obtained using the API of SHS. To enable researchers to experiment with not only harmonic but also rhythmic and timbral characteristics, it includes a large set of pre-extracted features along with the metadata that is linked to the composition and performance IDs used in SHS. Therefore, even though the audio files are not available, researchers can use the detailed metadata to recover the tracks themselves. Along with the benchmark set, the authors also released a framework called "acoss" for feature extraction and benchmarking designed for VI. It includes feature extraction functions with the hyperparameters used for preparing Da-TACOS and open-source implementations for seven VI systems. Furthermore, a training set for Da-TACOS was recently released, containing a training subset with 83,904 tracks (14,499 cliques), and a validation subset with 14,000 tracks (3,500 cliques).

**SHS-5+ & SHS-4- —** The last dataset we introduce in this section is SHS-5+ & SHS-4- (Doras & Peeters, 2019). It includes a training subset, SHS-5+, with 62,311 tracks in 7,460 cliques, and all the cliques have at least five versions each (hence the name). The test subset, SHS-4-, includes 48,483 tracks in 19,445 cliques, with two to four versions per clique, and it is the largest benchmark set for VI to date. Neither subset includes noise tracks. The version annotations are obtained using the SHS API, and the dataset includes a large set of pre-extracted features, including CQT and melodic representations.

### 2.5.2 Evaluation metrics

As previously introduced, VI systems aim to model the shared information between versions of the same underlying composition in order to provide a similarity score. The ability of a system to correctly assess the similarities between a query track and a corpus of tracks is usually evaluated by metrics that operate on a ranked sequence of results. Such a ranked sequence is obtained by first estimating similarity scores between a query and all the tracks in a corpus and then sorting the items in the corpus with respect to their similarities to the query. Below, we introduce the set of metrics typically used in VI.

**Precision and recall —** Being perhaps the most typical metrics in information retrieval, precision and recall provide rank-independent measures of system performance, which means that the order of the items does not affect the outcome. Precision gives the ratio of the retrieved items that are relevant[22] to all the retrieved ones (in other words, the accuracy of the system's predictions). Recall, on the other hand, gives the ratio of the retrieved items that are relevant to all the relevant items in the corpus (in other words, how well the system finds the relevant items). In VI, they are typically computed as Precision@$K$ (Prec@$K$) or Recall@$K$ (R@$K$) at a cut-off rank $K = \{1,5,10\}$, meaning only the first $K$ results are considered. Note that although precision and recall are rank-independent (i.e., the order of the items does not matter), Prec@$K$ and R@$K$ require a cut-off rank by definition and can be considered as rank-aware (i.e., the items have to be placed below rank $K$).

**Mean average precision —** Although precision and recall are common metrics, their rank-independent characteristics are not useful for tasks where the order of the retrieved items is important. A possible alternative for considering the ranks of the results is mean average precision (MAP). For this, average precision (AP) scores for all queries are computed and averaged. AP for a single query is obtained by averaging Prec@$K$ scores over all $K$ where a relevant item is returned, which makes AP a rank-aware metric in contrast to precision. Therefore, MAP assesses not only the number of relevant items in the results but also their ranks.

**Mean reciprocal rank —** Another rank-aware metric for assessing system performances is mean reciprocal rank (MRR). It is the average of reciprocal rank scores obtained for all the queries, where reciprocal rank is the multiplicative inverse of the rank of the first relevant item. Therefore, this metric is more appropriate for cases where either there is only one relevant item in the corpus, or if only the position of the first relevant item is important. Note that when there is a single relevant item in the corpus, MRR is equal to MAP.

**Mean rank of the first relevant item —** The last metric we introduce is the mean rank of the first relevant item (MR1). As MRR, this metric also uses only the first relevant item, and the only difference between them is the multiplicative inverse function in MRR. However, MR1 may be easier to interpret as ranks are taken directly. Note that the scale of differences between MR1 scores are the same everywhere, but such differences between MRR scores are scaled-down when higher ranks are considered. For example, when comparing two cases where the first relevant items have ranks 1 and 11, the reciprocal ranks are 1.00 and 0.09, respectively. On the other hand, when comparing two cases where the ranks of the first relevant items are 40 and 50, the reciprocal ranks are 0.025 and 0.020, respectively.

[22]Here, we use the term "relevant" to denote the items in the corpus that are versions of the query.

Chapter 3

# Toward Data-driven Version Identification

## 3.1 Introduction

When we started to work on the research presented in this dissertation, VI had been an active research field for almost two decades. There were many existing algorithms and datasets, widely accepted evaluation methodologies, and some promising ideas for further improvements. The "audio cover song identification" contest in MIREX had been running for more than a decade. There were two doctoral dissertations dedicated to VI (Serrà, 2011; Osmalskyj, 2017).

Although the state of VI research described above suggests that there was enough interest and consistent development in the field, certain paradigm shifts that were happening in the overall landscape of MIR were also influencing the VI research. The fact that MIR applications were getting commercialized more and more each year resulted in a shift toward data-driven and scalable systems for many tasks. The algorithms that had been tested on "toy datasets" started to fail in terms of providing the developers and users a realistic outlook on the performance in large-scale use cases. Therefore, the community started to adopt such data-driven and scalable solutions (specifically, deep learning–based ones) and to curate evaluation datasets that go beyond only a few hundreds of examples.

Following this trend from the general MIR landscape, we believed that the next generation of VI systems had to adopt a data-driven paradigm if they were ever to overcome the accuracy–scalability trade-off that previous VI systems had been struggling with. That is, the systems that performed the best (i.e., the local alignment–based ones) were the slowest ones in terms of computation times, and the fast systems (i.e., embedding-based ones) were consistently being outperformed by their slower alternatives. The pioneering deep learning–based VI systems showed that they were strong contenders to be the "holy grail" of VI systems by being both accurate and scalable at the same time; however, their generic model architectures and training methods mostly lacked any VI-related domain knowledge that the community had established throughout the years (e.g., techniques against common transformations between versions). Consequently, although being drastically faster compared with their conventional counterparts, such "deep" VI systems could not catch up with the performance of the local alignment–based methods yet.

Based on this, a natural starting point for this dissertation is to investigate ways to incorporate domain knowledge into deep VI systems. However, before delving into the machine learning part of our research, we first aim to address some other issues with the state of VI. For this, we identify four main points that the current VI research lacks: (1) widely accepted training and evaluation datasets, (2) a large-scale quantitative analysis on common changes between version pairs, (3) reproducible implementations of state-of-the-art algorithms for benchmarking, and (4) an analysis of the recent, data-driven input representations. While working on such issues is not crucial for developing novel VI systems, we consider them as a preliminary step that lays the foundation on which we build our research. This chapter, based on Yesiler et al. (2019), introduces the four aforementioned issues and our efforts to address each of them.

## 3.2 Curating a dataset

Data is a crucial part of machine learning systems and is often necessary for both model development and evaluation phases. Hence, our first step toward data-driven VI consists of curating a dataset that would enable us to develop novel VI systems and benchmarking existing systems to create baselines.

Although finding data of any modality (e.g., image, text, or audio) has become fairly easy thanks to the world wide web, finding labeled data is still not straightforward. Some companies have devoted substantial funds to annotate musical recordings in terms of various characteristics (e.g., the music genome project[23]), but the process of labeling is quite expensive for most academic, and even for some industrial, research. However, VI has a certain advantage over other MIR tasks in terms of resources for collecting labeled data. Since the early 2000s, there exist many user-driven websites that contain information about versions of existing musical works. Among them, SecondHandSongs[24] (SHS) is the biggest and the most famous one among the VI researchers. The version annotations are user-generated and verified by the website's editors, which assures that the data is of high quality. Each entry on the website includes track titles, artist names, version relationships, and other editorial metadata, along with YouTube and/or Spotify links for a high percentage of entries. The data on the website is licensed under Creative Commons BY-NC 3.0[25].

[23]https://www.pandora.com/about/mgp
[24]https://secondhandsongs.com/
[25]https://creativecommons.org/licenses/by-nc/3.0/

| Feature | Subset | | | Extraction library |
|---|---|---|---|---|
| | Training | Benchmark | Version analysis | |
| HPCP | ✓ | ✓ | ✓ | Essentia (Bogdanov et al., 2013) |
| MFCC | ✓ | ✓ | ✓ | Essentia (Bogdanov et al., 2013) |
| Key | ✓ | ✓ | ✓ | Essentia (Bogdanov et al., 2013) |
| CENS | ✓ | ✓ | ✓ | librosa (McFee et al., 2015) |
| Beat onsets | ✓ | ✓ | ✓ | madmom (Böck et al., 2016) |
| Tempo | ✓ | ✓ | ✓ | madmom (Böck et al., 2016) |
| cremaPCP | ✓ | ✓ | ✓ | crema (McFee & Bello, 2017) |
| Auto-tags | | | ✓ | CRNN (Choi et al., 2017) |

**Table 3.1:** List of features provided in each subset of Da-TACOS along with the software libraries used to extract them.

Using the SHS API, we collected version annotations and other editorial metadata for 112,904 tracks in 20,999 cliques, which became our main source of data for our research. We further split this data into several subsets for various purposes: (1) a "version analysis" subset for the experiments quantifying the frequency and extent of musical changes between versions (see Section 3.3); (2) a benchmark subset for evaluating VI systems (see Section 3.4); and (3) training and validation subsets for developing our deep learning–based models (see Chapters 4, 5, and 6). Using a set of open-source MIR libraries, we extracted a wide range of audio descriptors (see Table 3.1) and made them publicly available along with editorial metadata under Creative Commons BY-NC-SA 4.0 license[26]. We named this collection "Da-TACOS," which stands for "a dataset for cover song identification and understanding." To this day, it is the largest dataset curated for VI. More information on the dataset can be found in Appendix C.

## 3.3 Quantitative analysis of modifiable musical characteristics

Previous VI research repeatedly showed that having explicit mechanisms to obtain invariances against certain musical characteristics (e.g., key, tempo, and structure) results in considerable improvements to system performance. As shown in Figure 1.2, such characteristics can be grouped into nine categories. Although it is fairly straightforward to find examples of version pairs where differences in one or a few of those characteristics can be observed, there has not been any large-scale analysis that investigates how often and how drastically such differences occur. To address this, we now analyze a large collection of version pairs to quantify the frequency and extent of changes in five musical characteristics: key, tempo, timing, structure, and semantic aspects. With this, we aim to determine which characteristics to focus on handling

[26]40https://creativecommons.org/licenses/by-nc-sa/4.0/

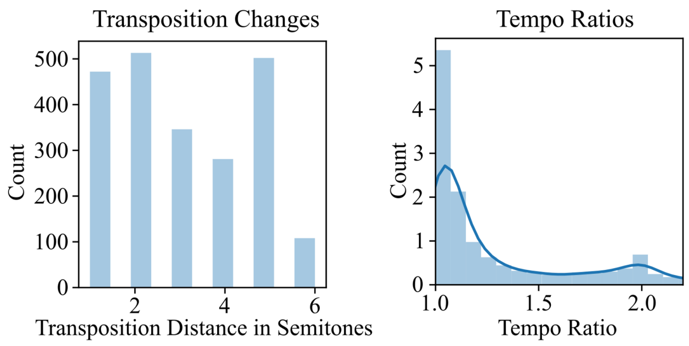


**Figure 3.1:** (Left) Distribution of changes in semitones between key estimates for version pairs with a reported key change. (Right) Distribution of tempo ratios between version pairs.

while developing our data-driven VI system.

### 3.3.1 Overview

We perform the analysis using the "version analysis" subset of Da-TACOS, which contains 5,000 version pairs in 5,000 unique cliques. Quantifying the key and tempo changes between pairs of versions is relatively straightforward using open-source MIR software libraries. However, to analyze the changes in timing, structure, and semantic aspects (e.g., instrumentation and genre), we devise custom distance measures. Apart from quantifying the changes in those characteristics between version pairs, we also obtain distances from those measures for non-version (i.e., unrelated) pairs. With this, we aim to show that the distances between unrelated pairs are higher than those between version pairs, which can be considered as a confirmation that our custom measures are appropriate[27]. We do this by forming distributions using the obtained distances for version and non-version pairs. To quantify the extent to which the distributions for versions and non-versions differ, we perform the two-sample Kolmogorov-Smirnov (KS) test (Massey, 1951) and report the resulting score along with its associated $p$-value, which indicates the statistical significance of the difference between the two distributions.

### 3.3.2 Analysis details

**Key —** Using a key estimation algorithm (Gómez, 2006), we consider all version pairs exceeding a key estimation confidence of 0.75, which are 4,288 pairs. Among these, 69.3% are reportedly in a different key. The distribution of semitone shifts between version pairs can be seen in the left of Figure 3.1. Thus, the use of techniques for handling transpositions between versions (see Section 2.3.2) is justified. One limitation of our analysis is that the key estimation algorithm reports a single estimate

[27]Although we expect changes in musical characteristics between version pairs, we expect such pairs to show more similar characteristics than unrelated pairs of tracks.

that is either major or minor. Under this scheme, the used algorithm reports that 17.5% of the pairs shift from major to minor keys. However, manually checking, we find that many of these examples are in modes beyond major and minor. To have a more fine-grained analysis to determine the changes beyond simple transpositions, more sophisticated algorithms or experts would be needed.

**Tempo —** We then examine tempo ratios between version pairs by picking out the tempo with the maximum confidence using a state-of-the-art tempo estimator (Böck et al., 2015; see the right of Figure 3.1). There is a slight peak around 2, which is likely due to “octave errors” from pieces that can be subdivided into 4/4. Beyond that, the changes in tempo that correspond to the first, second, and third quartiles are 1.03x, 1.11x, and 1.53x, respectively. These results suggest that half of the pairs are quite stable in terms of tempo; however, there are a fair number of pairs with a considerable change in tempo between the second and third quartiles. In conclusion, our analysis confirms that tempo stands out as a factor that needs to be considered while building VI systems.

**Structure —** A notably successful approach for analyzing music structure is using the eigenvectors of the graph Laplacian, or “spectral clustering” (McFee & Ellis, 2014). While we can assess the results obtained from this technique by comparing them to human-annotated labels for a particular set of tracks (McFee & Bello, 2017), using those eigenvectors to estimate the structural similarities between pairs of versions is not straightforward. Instead, we form vectors for each track using the eigenvalues of the graph Laplacian, and we compare the similarities of such vectors, and thus, the structural similarities between tracks, using Euclidean distance. This technique was referred to as “ShapeDNA” in the context of 3D shape analysis of triangle meshes (Reuter et al., 2006). For our analysis, we use feature-fused self-similarity matrices (Tralie & McFee, 2019) that we downsample to a common dimension of $256 \times 256$, and 30 eigenvalues of a random walk Laplacian to form the vectors per track.

Figure 3.2 shows self-similarity matrices and ShapeDNAs of a pair of versions and a third, unrelated track. We observe that the self-similarity matrices of the version pair look much more similar to each other compared to that of the third track. Supporting that observation, their ShapeDNAs demonstrate a similar outcome where the vectors of the version pairs appear close while being considerably different from that of the third track. Based on this, we hypothesize that computing similarities between ShapeDNAs is an appropriate proxy for estimating structural similarities between tracks. We then form distributions of ShapeDNA distances for version and non-version pairs (see Figure 3.3). We see that although the obtained distances are higher for unrelated pairs of tracks, structural differences exist between versions, which is indicated by the distance scores. Lastly, the KS score between the two distributions is 0.22 ($p < 0.01$), confirming that there are more drastic structure variations between pairs of unrelated tracks while versions, although still incorporating differences, may

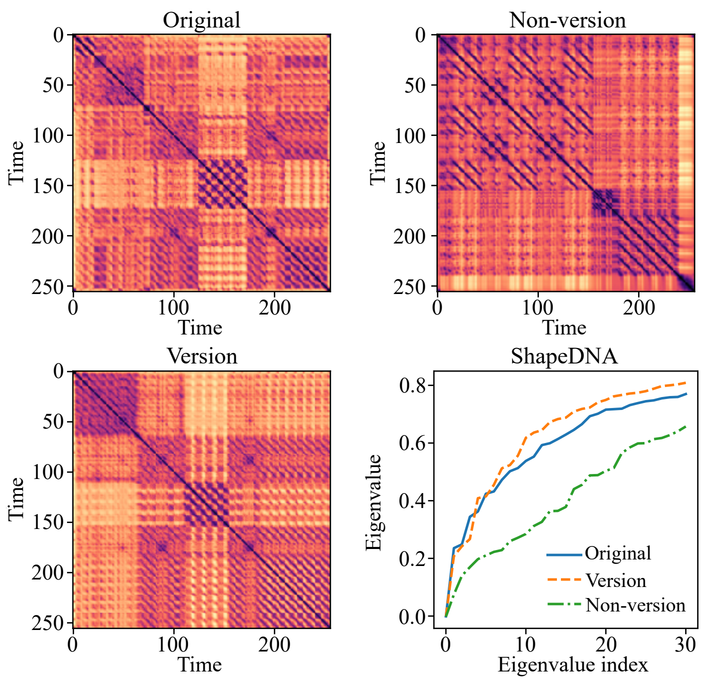


**Figure 3.2:** An example of fused similarity matrices of "The Wizard" by Uriah Heep (upper left), a version by Blind Guardian (lower left), and "Million Pieces" by The Piano Tribute Players (upper right), which is unrelated to the other two tracks. The corresponding ShapeDNAs are shown in the lower right.

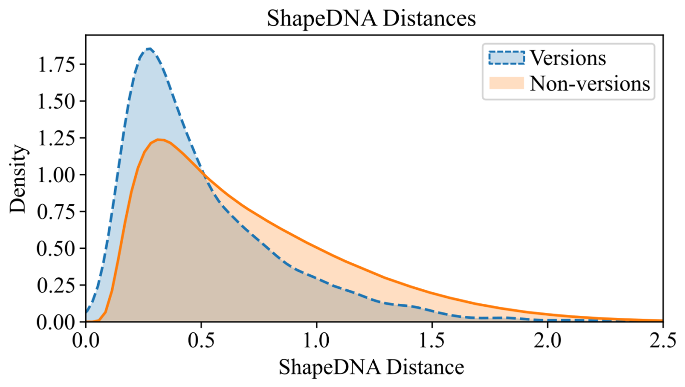


**Figure 3.3:** Distributions of ShapeDNA differences between pairs of tracks as a means of assessing structural changes.

have more in common regarding overall structure.

**Timing —** We now analyze variations in timing, which we define as local changes in tempo over time. For this, we first extract $N$ beat onset estimates $B(t)$, where $t = 1,2,\dots,N$ using the algorithm proposed by Krebs et al. (2015), down to a resolution

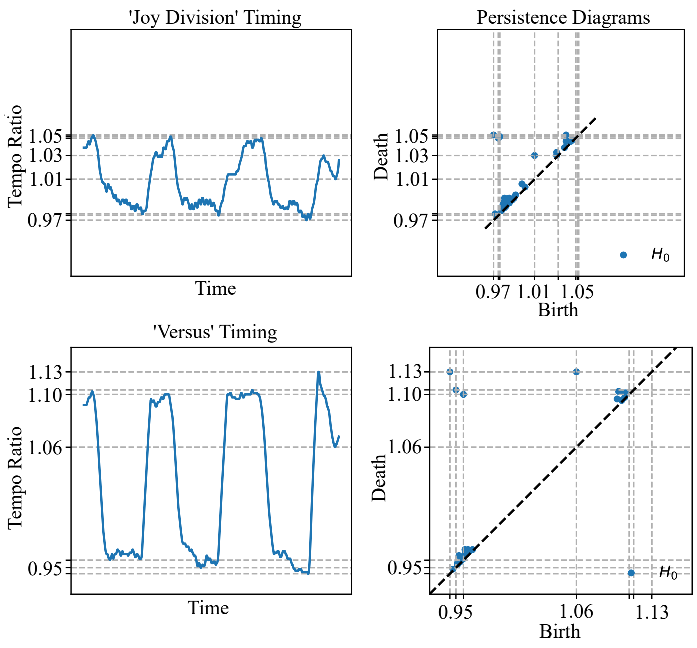


**Figure 3.4:** An example of $R(t)$ estimates and their associated persistence diagrams for the track "24 Hours" by Joy Division and Versus. Both tracks speed up in the chorus and slow down in the verse; therefore, they each contain several local minimums with high persistence that are born during the verses. They each also contain some low amplitude wobbling which shows up as dots near the diagonal.

of 10 ms. We then obtain unit-less local tempo estimates by convolving $B(t)$ with a Gaussian derivative $G(t) = -te^{-t^2/2}$ to obtain $B'(t) = B(t) * G(t)$ and smooth out noise by applying a sliding window average of width 20. Finally, we divide $B'(t)$ by its median to obtain a relative, tempo-normalized local tempo deviation $R(t)$, where $R(t) > 1$ indicates that a track speeds up locally around time $t$, and $R(t) < 1$ indicates a slowing down.

The left column of Figure 3.4 shows $R(t)$ for two versions of the same track. Note the multiscale features of $R(t)$, from small wobbles to large changes that persist over a section. To capture all scales in one distance measure that can tolerate missing beats and structural changes (e.g., added/deleted sections), we use "lower star filtration," a watershed method from topological data analysis (Edelsbrunner & Harer, 2010). It summarizes a time series in a "persistence diagram"[28] (PD). It was previously used, for instance, on quantifying driving behavior by analyzing the speed time series of drivers (Rouse et al., 2015).

The right column of Figure 3.4 shows PDs for the $R(t)$ for an example version pair,

[28]A multiset of points whose x-coordinates correspond to local minimums where pools of water form as the water rises from bottom to top ("birth events") and whose y-coordinates correspond to local maximums paired to such minimums where two pools merge together ("death events")

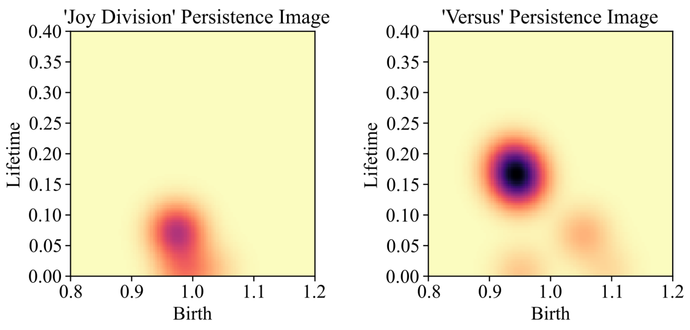


**Figure 3.5:** Persistence images corresponding to the diagrams in Figure 3.4. The Versus version contains larger scale wobbles, leading to blobs further up on the persistence axis. However, they both contain also smaller scale wobbles.

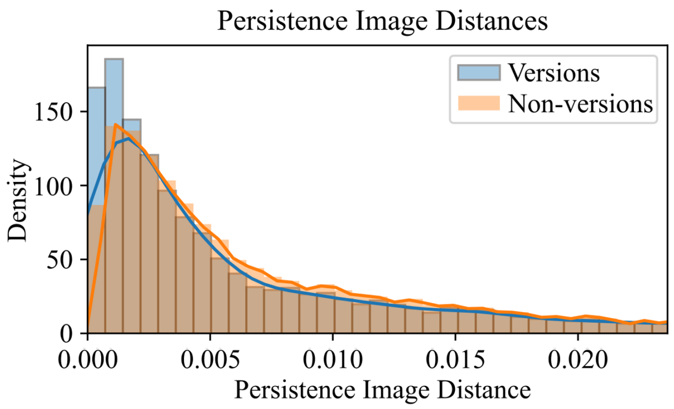


**Figure 3.6:** Distribution of persistence image distances of lower star filtrations of relative tempo functions between pairs of tracks.

with the birth and death values of the points with the four largest "persistence" (death-birth) marked. To compare PDs of two different tracks, we use persistence images (Adams et al., 2017; see Figure 3.5). They transform a PD into birth–persistence space and place a Gaussian over each point, whose magnitude is proportional to the persistence. Figure 3.6 shows the distributions of Euclidean distances between persistence images for version and non-version pairs. Although the distributions look quite similar, the KS score is 0.095 ($p < 0.01$), indicating that although variations in timing are more drastic for unrelated tracks than for version pairs, such variations are also observed between versions as the mode of the related distribution is greater than 0, which is also supported by Figure 3.4 where one track speeds up more in the chorus relative to the other.

**Semantic aspects —** To analyze the similarities regarding the semantic aspects of the tracks (e.g., mood, instrumentation, and genre) without explicitly defining them, we use the auto-tagging model of Choi et al. (2017), which uses log-mel spectrograms as

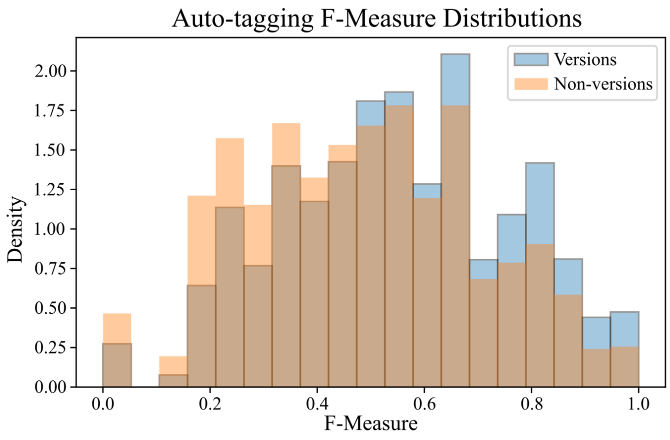


**Figure 3.7:** Distributions of F-measures for version and non-version pairs regarding the higher-level semantic aspects.

input to return a set of tags that qualitatively describe a track. Since the model returns many tags with low confidence, we only consider the ones that are in the 90th percentile over all confidence scores, which corresponds to a value of 0.062. If $R_1$ is the fraction of the tags obtained for track A that are also in the set of tags for track B, and $R_2$ is the fraction of tags obtained for track B that are also in the set of tags for track A, then the F-measure between two tracks is defined as $2R_1R_2/(R_1+R_2)$, which is 1 if they are in complete agreement and 0 if they have nothing in common. Figure 3.7 shows the distribution of F-measures between version and non-version pairs. While the distributions are overall quite similar, the F-measures are skewed slightly lower for non-versions. The KS score between the two distributions is 0.118 ($p < 0.01$), indicating that these two distributions are statistically different and, therefore, stylistic changes occur more drastically between unrelated tracks compared with between version pairs. However, the fact that the mode of the F-measure distribution for version pairs is far from 1 suggests the importance of variations in such semantic aspects between versions.

## 3.4 Benchmarking existing systems

For many years, the lack of a unified and widely accepted evaluation setting remained one of the main limitations of VI research. Firstly, the majority of existing VI systems did not have any open-source implementations, with only a few exceptions. This means that researchers who wanted to benchmark their new approaches developed on a new dataset had to first implement a large set of existing systems. Along with the fact that such systems may incorporate many small details that are not described in the corresponding research papers, it is known that such reimplementations generally include differences that can degrade the quality of the benchmarking. Secondly,

| | MAP | MR1 |
|---|---|---|
| *Individual systems* | | |
| SSM (Tralie & Bendich, 2015) | 0.096 | 434 |
| 2DFTM (Bertin-Mahieux & Ellis, 2012) | 0.126 | 207 |
| SiMPle (Silva et al., 2016) | 0.165 | 358 |
| Dmax (Chen et al., 2018a) | 0.348 | 167 |
| Qmax (Serrà et al., 2009a) | 0.385 | 136 |
| Qmax* (Serrà et al., 2009b) | 0.390 | 131 |
| *Ensemble systems* | | |
| SNF (Qmax + Dmax) (Chen et al., 2018a) | 0.471 | 161 |

**Table 3.2:** Performance statistics for the baseline systems using HPCP features as input.

the then-available public datasets all had major drawbacks that pushed researchers to use proprietary or newly curated data for system evaluation. Among such drawbacks were the insufficient dataset size (e.g., only a few hundreds of tracks), the limited set of shared features (e.g., only PCP feature variants), the use of proprietary algorithms for extracting such features (e.g., the EchoNest API), and so on. Naturally, addressing this issue of not having a proper benchmarking setting becomes a priority for us before developing any system. Considering both sides of the problem, we curated a benchmark dataset and implemented seven state-of-the-art VI systems all of which were made publicly available for the community.

The benchmark dataset, which is a subset of Da-TACOS, includes a total of 15,000 tracks: 13,000 tracks in 1,000 cliques with 13 tracks each, and 2,000 tracks that do not belong to any other clique, acting as noise in the data (see Section 2.5.1). We used open-source MIR libraries to extract a wide range of features that are commonly used in VI literature. The code for the feature extraction and the parameters we used are publicly available in the project repository[29] (see Appendix C).

As for the open-source implementations of VI systems, we consider seven systems that incorporate different techniques and provide different approaches to VI. While systems like the ones proposed by Bertin-Mahieux & Ellis (2012) and Silva et al. (2016) put more emphasis on the scalability aspect by focusing on reduced computation times, other systems proposed by Serrà et al. (2009a) and Chen et al. (2018a) incorporate local alignment–based algorithms that provide slow but high-performance results. By having reference implementations for such systems, we aim to encourage VI researchers to conduct comprehensive benchmarking when developing and evaluating novel systems.

Table 3.2 presents the initial benchmarking results of the implemented systems on the Da-TACOS benchmark subset. The results are consistent with other benchmarks like the MIREX competition. Qmax* outperforms all the individual systems by consider-

[29] https://github.com/furkanyesiler/acoss

able margins. However, the ensemble system proposed by Chen et al. (2018a), which uses the similarity network fusion (SNF) technique to combine networks of pairwise distances obtained with Qmax and Dmax algorithms (see Section 2.4.3) sets a strong baseline by outperforming all the individual systems including Qmax*.

## 3.5 Search for data-driven input features

In VI literature, the most popular input features have been the PCP variants (see Section 2.3.1). Although having been developed in the early years of MIR research, they have proven their usefulness in many tasks over the years. However, seeing that the latest developments in MIR point to the success of data-driven methods over the knowledge-driven ones, we now aim to analyze whether some of the recently developed data-driven features would bring any performance gains in VI.

Focusing on PCP variants, our first candidate is the "deep-chroma" feature proposed by Korzeniowski & Widmer (2016), which was shown by Silva et al. (2018) to outperform CENS features. Deep-chroma features are extracted using a multilayer perceptron (MLP) model, which is trained for the automatic chord recognition (ACR) task. Our second candidate is the pitch class features of the crema model, proposed by McFee & Bello (2017). Crema is a convolutional recurrent neural network that was also trained for the ACR task. However, by adopting a structured prediction strategy, the crema model estimates the root, the bass, and the pitch classes for each frame, which are later combined to output a single chord. Crema model is designed to tackle the large-vocabulary ACR task, which considers 14 chord qualities as opposed to 2 qualities (i.e., major and minor) considered by the deep-chroma model.

After a set of preliminary experiments, we chose the pitch class features obtained from the crema model, which we call cremaPCP in short, over the deep-chroma features. For the main evaluation, we compare the performances of six state-of-the-art VI systems with HPCP and cremaPCP features as input. As seen in Table 3.3, cremaPCP constantly outperforms HPCP for all the considered systems with relative increases varying between 13% and 118% in terms of MAP. These results suggest that cremaPCP is a promising choice as an input representation for the first VI system we develop (see Chapter 4).

## 3.6 Conclusion

In this chapter, we have described our efforts toward developing data-driven VI systems. Firstly, we have discussed the state of VI research when we began working on this dissertation, along with the developments in the overall MIR landscape. We have explained our intuitions regarding why a data-driven paradigm must be adopted to tackle the struggles related to the accuracy–scalability trade-off. We have then identified four main issues to address before developing our VI systems. Our efforts toward

| | Input | MAP | MR1 |
|---|---|---|---|
| *Individual systems* | | | |
| SSM (Tralie & Bendich, 2015) | M | 0.096 | 434 |
| 2DFTM (Bertin-Mahieux & Ellis, 2012) | H | 0.126 | 207 |
| | C | 0.275 | 155 |
| SiMPle (Silva et al., 2016) | H | 0.165 | 358 |
| | C | 0.332 | 142 |
| Dmax (Chen et al., 2018a) | H | 0.348 | 167 |
| | C | 0.398 | 167 |
| Qmax (Serrà et al., 2009a) | H | 0.385 | 136 |
| | C | 0.437 | 138 |
| Qmax* (Serrà et al., 2009b) | H | 0.390 | 131 |
| | C | 0.445 | 130 |
| *Ensemble systems* | | | |
| SNF (Qmax & Dmax) (Chen et al., 2018a) | H | 0.471 | 161 |
| | C | 0.533 | 139 |

**Table 3.3:** Performance statistics for the baseline systems implemented comparing HPCP and cremaPCP features. H stands for HPCP, C for cremaPCP and M for MFCC.

such issues resulted in (1) the release of the largest dataset for VI, which contains predetermined training and evaluation subsets for data-driven model development; (2) the first large-scale quantitative analysis on the frequency and extent of changes in musical characteristics between versions, which has provided useful insights on the importance of achieving invariances against certain variations when developing a VI system; (3) a framework with reference implementations of a selected set of state-of-the-art VI systems, which can facilitate reproducibility and benchmarking efforts for VI researchers; and (4) a preliminary analysis that compares hand-crafted and data-driven input features for VI, which has suggested performance improvements achieved by replacing hand-crafted features with their data-driven counterparts. The insights, tools, and datasets that we have described in this chapter constitute the foundation of our methods described in the following chapters of this dissertation.

# Chapter 4
# Musically Motivated Version Embeddings

## 4.1 Introduction

In this chapter, we describe our first VI system, which lays the foundation for the rest of our research. As mentioned in Chapter 3, the latest developments in MIR when we started to work on the research presented in this dissertation suggested that the next generation of VI systems had to build upon data-driven strategies. Therefore, using the findings from the preliminary work introduced in Chapter 3, we now propose a data-driven VI system that we call MOVE, which stands for "musically motivated version embeddings." MOVE achieves state-of-the-art performance on two publicly available benchmark datasets by integrating domain knowledge into a data-driven system design. Compared with existing alternatives, it demonstrates a substantial improvement in accuracy while being suitable for large-scale use cases.

VI systems that encode full tracks into compact embedding vectors bring drastic benefits in terms of scalability, as they allow for lightweight similarity estimation between items. However, the pioneering systems that used such embedding-based approaches had issues with finding effective ways for the encoding process. The later data-driven systems addressed those issues by taking advantage of the developments in other data-driven applications, mainly using more powerful network architectures and better optimization schemes. However, such systems mostly included general components that were task-agnostic and, although improved upon their predecessors, reached only a certain level of performance. In an effort to outperform that first generation of data-driven VI systems, in this work, we put an emphasis on incorporating domain knowledge into data-driven workflows. Based on the results presented in Section 3.3, we employ explicit strategies to handle common variations observed between versions. With an ablation study, we justify each of such strategies (see Section 4.3.3).

Apart from combining domain knowledge with data-driven systems, another key aspect of MOVE is that it employs a contrastive learning objective for optimization. In

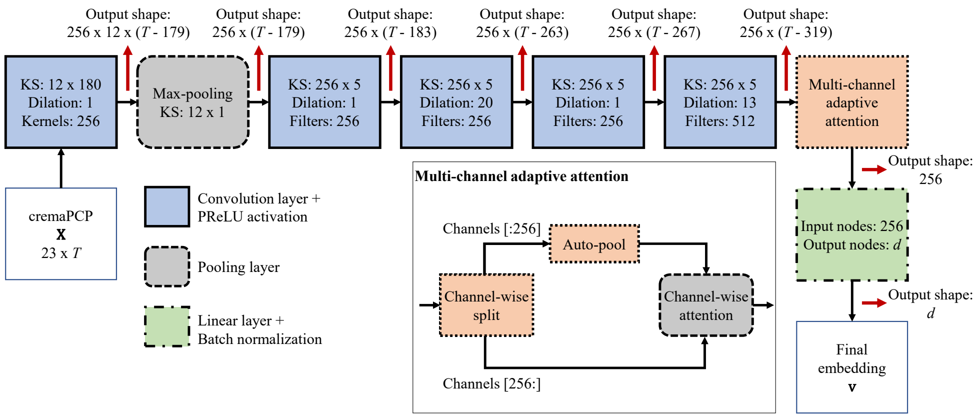


**Figure 4.1:** Block diagram of MOVE's architecture. KS denotes kernel size.

supervised learning, classification-based training is by far the most popular approach; however, the way the optimization process is formulated may create issues when there are a large number of classes[30]. By following a contrastive learning paradigm, we aim to avoid any potential issues regarding the number of classes and the number of items per class in our training dataset. We further discuss such potential issues and the advantages of contrastive learning over classification-based training in Section 4.2.3.

This chapter is based on Yesiler et al. (2020a)[31].

## 4.2 Methods

In this section, we describe and motivate our design decisions regarding the individual components and the training strategies of our system. Overall, MOVE takes cremaPCP features as input and processes them through five convolutional layers, interleaved with two pooling layers and multiple nonlinearity functions to encode input tracks into single embedding vectors of a fixed length. It has explicit modules to handle pitch transpositions and structural changes. We use a metric learning approach for training by employing a triplet loss function. We also propose VI-specific data augmentation strategies that aim to make MOVE more robust against changes in pitch transpositions, tempo, and timing.

### 4.2.1 Input representation

Based on the preliminary results presented in Section 3.5, we choose cremaPCP as the input representation of MOVE (see Section 3.5 for details). We use the pretrained

[30]Note that ImageNet, a popular and large-scale benchmark dataset for computer vision, has only 1,000 categories while the Da-TACOS training set includes 14,499.

[31]© 2021 IEEE. Reprinted, with permission, from Yesiler, F., Serrà, J., & Gómez, E. (2020). Accurate and scalable version identification using musically-motivated embeddings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 21–25.

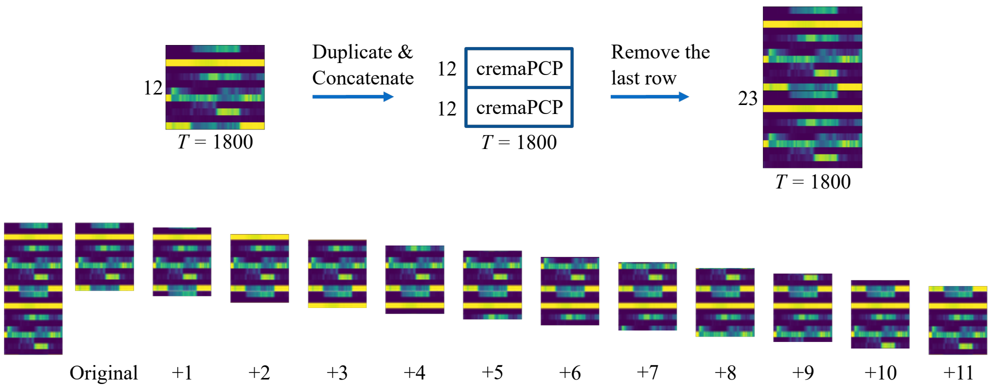


**Figure 4.2:** The preprocessing of the input features for the transposition invariance module (top). The resulting features of shape $23 \times T$ includes all the possible transpositions of the original input (bottom).

model available at the related repository[32]. We denote the cremaPCP features for a track by $\mathbf{X} \in [0,1]^{12\times T}$, where $T$ is the number of frames using a hop length of 93 ms. For training, we take random patches of $T = 1800$ frames after applying data augmentation (see Section 4.2.3) to entire tracks. At inference time, we use entire tracks without picking random patches of a particular length (preliminary experiments showed that the temporal summarization strategy described below was also effective with entire tracks at inference time).

### 4.2.2 Network architecture

MOVE consists of five convolutional blocks with parametric-ReLU (PReLU) activation functions and no padding, interleaved by two different pooling layers (Figure 4.1). A linear layer followed by nonparametric batch normalization produces the final embeddings for each track. With the best setup, the total number of parameters is 6.3 M. We now motivate and present the key components of MOVE.

**Transposition invariance module —** We have shown in Section 3.3 that key changes are quite common between version pairs. Consequently, we use an explicit module for achieving a certain degree of invariance against such changes, which combines a convolutional and a max-pooling layer. Following the strategy proposed by Xu et al. (2018), we change the shape of the cremaPCP inputs $\mathbf{X}$ from $12\times T$ to $23\times T$ by concatenating two copies of $\mathbf{X}$ in the pitch class dimension and removing the last pitch class. The resulting representation $\hat{\mathbf{X}}$ contains all possible transpositions of the original input $\mathbf{X}$ (see Figure 4.2). Then, the first set of convolution kernels $\mathsf{L}^{(1)} = \{\mathbf{L}_1^{(1)}, \mathbf{L}_2^{(1)}, ..., \mathbf{L}_{256}^{(1)}\}$, where $\mathbf{L}_n^{(1)} \in \mathbb{R}^{12\times 180}$, traverse the modified input, going through all possible transpositions in the pitch dimension and outputting the internal representation $\mathcal{H}^{(1)} \in \mathbb{R}^{256\times 12\times (T-179)}$. After a PReLU function, the sub-

[32] https://github.com/bmcfee/crema (version 0.1.0)

sequent max-pooling operation $M : \mathbb{R}^{256\times12\times(T-179)} \to \mathbb{R}^{256\times(T-179)}$ keeps the transposition with the highest activation value, which results in the internal representation $\check{\mathbf{H}}^{(1)} \in \mathbb{R}^{256\times(T-179)}$.

**Expanding the receptive field —** The four convolutional blocks applied to $\check{\mathbf{H}}^{(1)}$ (i.e., $\mathsf{L}^{(2)} = \{\mathbf{L}_1^{(2)}, \mathbf{L}_2^{(2)}, ..., \mathbf{L}_{256}^{(2)}\}$, $\mathsf{L}^{(3)} = \{\mathbf{L}_1^{(3)}, \mathbf{L}_2^{(3)}, ..., \mathbf{L}_{256}^{(3)}\}$, $\mathsf{L}^{(4)} = \{\mathbf{L}_1^{(4)}, \mathbf{L}_2^{(4)}, ..., \mathbf{L}_{256}^{(4)}\}$, and $\mathsf{L}^{(5)} = \{\mathbf{L}_1^{(5)}, \mathbf{L}_2^{(5)}, ..., \mathbf{L}_{512}^{(5)}\}$, where $\mathbf{L}_n^{(2)}, \mathbf{L}_n^{(3)}, \mathbf{L}_n^{(4)}, \mathbf{L}_n^{(5)} \in \mathbb{R}^{256\times5}$) are used to encode higher-level information and to increase the receptive field of the model (Figure 4.1). On the one hand, with the layers that have a dilation rate of 1 (i.e., $\mathsf{L}^{(2)}$ and $\mathsf{L}^{(4)}$), we aim to encode higher-level nonlinearities while expanding the temporal context minimally. On the other hand, with the layers that have dilations 20 and 13 (i.e., $\mathsf{L}^{(3)}$ and $\mathsf{L}^{(5)}$, respectively), we swiftly increase the receptive field, which is 17 s per frame at $\check{\mathbf{H}}^{(1)}$, to approximately 30 s after $\mathsf{L}^{(5)}$. Notice that this temporal span could already be sufficient to detect musical versions, at least from a human perspective. However, to process an even larger time span, to be able to deal with different lengths $T$ at test time, and to deal with structural changes, we still perform an additional step.

**Summarizing the temporal content —** Following the idea that deep neural networks provide an opportunity to learn useful features rather than using hand-crafted ones, such models can be considered as having two parts: a feature extractor and a classifier/projection head, which, generally speaking, consist of convolutional blocks (e.g., convolutional layers, nonlinear activation functions, and normalization layers) and fully connected layers, respectively. Therefore, the modules that have been introduced above all fit into the "feature extractor" part of MOVE.

An ideal feature extractor for VI must handle a few important considerations. Firstly, having explicit or implicit strategies against common changes in musical characteristics is crucial for VI, which has been emphasized earlier. Secondly, VI is a task where (usually) entire tracks are compared for similarity estimation. This naturally requires a feature extractor that can deal with inputs of varying sizes. Note that this requirement can be ignored with practical solutions: to use a feature extractor that can process inputs of only a certain duration, one can randomly select patches of a fixed size (for inputs with a longer duration than the required patch size) or pad the input features to reach the required size (for inputs that are shorter than the required patch size). However, this brings another point to consider. In cases where we randomly select patches from a long input, we may miss important information that may be helpful for the task. For example, in some live performances, performers may add long, improvised solo sections that are not present in any other version of the same track. If the VI system randomly selects only those solo sections to process, it may be difficult to identify the relationship between version pairs. One must keep in mind that when processing tracks for VI, we often do not have any information about which parts of the tracks are more important or useful for the system. Therefore, designing a feature extractor that can process tracks of varying lengths is highly desirable for VI. A common way to do this is by summarizing or aggregating the information in

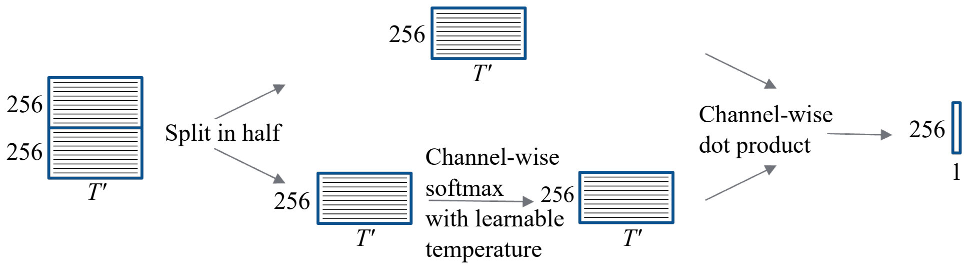


**Figure 4.3:** The multi-channel adaptive attention module.

the temporal dimension.

Previous techniques for summarizing the temporal information include average- and max-pooling operations. Although they provide a solution for processing inputs of varying sizes, they are not ideal for VI. When summarizing the temporal content, the average-pooling operation considers all time steps as equally important. However, this assumption does not hold well for many popular styles or genres. Generally, certain sections of tracks are more important than others: for example, the chorus, verse, or bridge sections may hold more information for us to detect version pairs than the intro or outro sections. The max-pooling operation, on the other hand, captures only the most "informative" time step, which is the time step with the highest activation value. Although the idea of identifying only the most important time step is more plausible than considering all the time steps as equally important, it may still be problematic in certain cases as it limits the source of information to only a single time step, and the gradient flow through that selected frame during backpropagation may be suboptimal.

Considering the shortcomings of the average- and max-pooling operations, we use a strategy that aims to let the network figure out which time steps are important and how important they are. With this, one may incorporate several time steps while aggregating information (as opposed to max-pooling) and at the same time indicate the relative importance of those time steps (as opposed to average-pooling). We call this solution multi-channel adaptive attention mechanism, which combines multi-channel temporal attention (Serrà et al., 2018) with auto-pool (McFee et al., 2018). The first idea is to let the network compute (and learn) the importance of each time step independently for each feature with a local attention–like mechanism (Serrà et al., 2018). The second idea is to apply a nonlinear, learnable pooling function that uses a scaling parameter before the softmax function (McFee et al., 2018) such that depending on the value of such parameter, the function pivots between average- and max-pooling. In practice, temporal summarization is done by calculating channel-wise attention weights, which correspond to the second half of the activation maps resulting from the last convolutional layer ($L^{(5)}$) $\mathbf{H}_a^{(5)} \in \mathbb{R}^{256\times(T-319)}$, where $a = 257, 258, ..., 512$, with the auto-pool function, and using the result to weigh the second half of the activation maps resulting from the same convolutional layer $\mathbf{H}_b^{(5)} \in \mathbb{R}^{256\times(T-319)}$, where

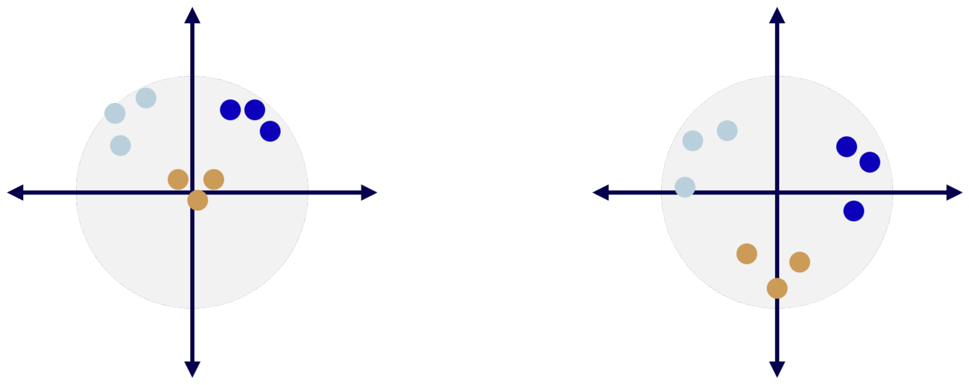


**Figure 4.4:** Two cases where items in the latent space are clustered well. Note that the half of the space from the left figure is unused.

$b = 1,2,...,256$ (see Figure 4.3). This corresponds to

$$\mathbf{h}_i^{(6)} = \sum_{t=1}^{T-319} \sigma(\alpha \mathbf{H}_{i+256}^{(5)}) \odot \mathbf{H}_i^{(5)}, \tag{4.1}$$

where $i = 1,2,...,256$, the sum is taken across the temporal dimension, $\sigma$ corresponds to the softmax function, $\alpha$ is a learnable parameter which we initialize to 0 (which makes the operation equivalent to average-pooling), and $\odot$ is the element-wise product. The result of this mechanism is a fixed-size feature vector $\mathbf{h}^{(6)} \in \mathbb{R}^{256}$, regardless of the duration $T$ of the input track.

**Standardizing embedding components —** For deep metric learning approaches using a triplet loss, it is important to consider the characteristics of the latent space where the embeddings lie, especially during training. Firstly, the volume of the latent space should be controlled: for instance, in the case where a margin hyperparameter is used in the loss function (e.g., the triplet loss), if the magnitude of the distances and the margin are disproportionate, the training process may not be able to structure the latent space in an effective way. Secondly, making sure that all the dimensions of the latent space being utilized similarly may improve the expressivity of the embeddings. Figure 4.4 shows two cases where, although the items are clustered well in both, half of the space on the left figure is not utilized, which may affect the inference performance when dealing with unseen data.

With these motivations in mind, we use nonparametric batch normalization after the linear layer that finalizes the encoding process. By doing so, we aim to obtain zero-mean and unit-variance components in our embeddings, yielding a statistically standardized latent space volume and similarly utilized latent dimensions because dimensions are treated i.i.d. This, together with dimension-normalized Euclidean distances, also helps us to develop some intuition regarding the loss values and the corresponding margin for the triplet loss.

### 4.2.3 Training strategy

In supervised learning, training models using a classification approach is the most common way. Many research datasets contain items that are categorized into distinct classes, which consequently encourages the development of classification-based training objectives or evaluation methods. In VI, straightforward ways of formulating the model training as a classification task are either by considering each clique as a distinct class (i.e., multiclass classification) or by labeling track pairs as versions or non-versions (i.e., binary classification; see Section 2.3.5.2).

An alternative to classification-based training is to use metric learning, which conceptually aims to learn linear or nonlinear distance functions that position semantically related items closer than unrelated items in a latent space. Although strong connections exist between certain types of classification- and metric learning–based training strategies and objective functions, we favored the latter while training MOVE for the reasons outlined below.

Firstly, from a theoretical perspective, the large (and practically unbounded) number of classes in VI makes it less suitable for classification-based training. To demonstrate this, we can consider the most common multiclass classification objective function: the negative log-likelihood, which is denoted as

$$\mathcal{L}_i^{\text{NLL}} = -\log\left(\frac{\exp(q_i^{(c^*)})}{\sum_{c\in C}\exp(q_i^{(c)})}\right), \tag{4.2}$$

where $c^*$ is the correct class for item $i$, $q_i^c$ is the logit for the class $c$ for item $i$, and $C$ is the set of all classes in the training set. The denominator of the softmax operation, which is necessary to scale the logits between 0 and 1 and to turn the logits into "probabilities," requires the summation of the exponentials of all the logits for all the classes for a single input. In the case where the number of classes is large (e.g., in the order of tens or hundreds of thousands), this computation becomes expensive from a resource point of view. Secondly, while the objective function tries to bring all items from the same class closer to each other, it pushes them away from all the other classes at the same time. Therefore, the quality of the solution is not only determined by how close the items of the same class are to each other but also by how far they are from the items of the other classes. Although this may be a desirable property, finding solutions that satisfy this may be difficult from an optimization perspective. Lastly, the number of items per class in a training dataset is an important factor for the success of classification-based training. In VI, it is common to have cliques of musical works with a small number of tracks (e.g., less than 5). Although data augmentation techniques can be used to increase the number of items per class (see below), alternative methods that work well with small cliques may avoid this constraint.

The common objective functions for metric learning (e.g., the triplet loss) address these issues by relaxing the training conditions. In particular, the triplet loss considers triplets of items (an anchor, a positive, and a negative) and aims to make the distance between the related items [i.e., the anchor ($\mathbf{v}_\mathrm{A}$) and the positive ($\mathbf{v}_\mathrm{P}$)] smaller than the distance between the unrelated items [i.e., the anchor ($\mathbf{v}_\mathrm{A}$) and the negative ($\mathbf{v}_\mathrm{N}$)] by a predetermined margin $m$. With this, the models only need to satisfy a relative distance condition, rather than considering the absolute distances between the items. Also, since only a triplet of items is needed to compute the loss, it is a much cheaper operation than computing the softmax. However, the main problem with the triplet loss is that the strategy for choosing which triplets to use for computing the loss function (and, therefore, training the model) has a drastic impact on the training performance, as we show in our ablation study (see Section 4.3.3).

Based on motivations outlined above, we use the triplet loss for training MOVE, which is denoted as

$$\mathcal{L}^{\text{Triplet}}_{\text{A, P, N}} = \max\left(E\left(\mathbf{v}_\mathrm{A}, \mathbf{v}_\mathrm{P}\right) - E\left(\mathbf{v}_\mathrm{A}, \mathbf{v}_\mathrm{N}\right) + m, 0\right), \tag{4.3}$$

using

$$E(\mathbf{v}_i, \mathbf{v}_j) = \frac{1}{d}\,\|\mathbf{v}_i - \mathbf{v}_j\|^2, \tag{4.4}$$

where $\|\ \|$ corresponds to the Euclidean norm and $\mathbf{v}$ denotes an embedding of size $d$ produced by the model. We now present our decisions regarding triplet mining, training data, data augmentation, and hyperparameters.

**Triplet mining —** As discussed in previous works that employ a triplet loss, the characteristics of the triplets in each mini-batch may have drastic effects on learning performance (Schroff et al., 2015; Hermans et al., 2017). For our model, we employ an online hard triplet mining strategy (Hermans et al., 2017). In our implementation, we choose 16 unique cliques and 4 tracks per clique, forming a mini-batch of 64. For the cliques that have less than 4 tracks, we choose among the already chosen tracks of the same clique. Within a mini-batch, we consider all the examples as anchors ($\mathbf{v}_\mathrm{A}$), and select the positive/negative example that has the maximum/minimum distance to the anchor ($\mathbf{v}_\mathrm{P}$ and $\mathbf{v}_\mathrm{N}$, respectively). Although Schroff et al. (2015) point out that the hardest examples may lead to local minima early in the training, our triplets can be considered "moderate" (Hermans et al., 2017), as they are selected only from the current mini-batch, and therefore, do not strictly correspond to the hardest triplets in the dataset. This presumably avoids the aforementioned local minima.

**Training data —** For training, we use the training and validation subsets of Da-TACOS (see Section 3.2), which include 14,499 cliques containing 83,905 tracks and 3,500 cliques containing 14,000 tracks, respectively. All audio files are encoded in MP3 format and their sample rate is 44.1 kHz.

**Data augmentation —** In order to enhance the learning of MOVE, we apply to each training example a data augmentation function specifically designed for VI. Considering the common transformations between version pairs, such function sequentially and independently applies pitch transposition, time stretching, and time warping with probabilities 1, 0.3, and 0.3, respectively. Transposition uses the octave-equivalent characteristics of PCP representations and randomly rolls $\mathbf{X}$ in the pitch dimension between 0 and 11 bins. Time stretching uses one-dimensional interpolations in the temporal domain, with a random factor between 0.7 and 1.5. Time warping consists of three mutually exclusive functions, which either silence, duplicate, or remove frames with probabilities 0.3, 0.4, and 0.3, respectively (silence corresponds to zeroing out the entire frame). Once selected, these functions are applied on a per-frame basis with a probability of 0.1, 0.15, and 0.1, respectively. All random numbers are sampled using a uniform distribution.

**Hyperparameters and optimization —** We train MOVE for 120 epochs with plain stochastic gradient descent (SGD), using an initial learning rate of 0.1 and decreasing it by a factor of 5 at epochs 80 and 100. An epoch is completed when our data loader goes through all possible cliques. However, an important detail to note is that we include the cliques with sizes between 6 and 9 twice, the ones with sizes between 10 and 13 three times, and the ones with sizes 14 or above four times. This is done to increase the probability of every track being introduced to the network at least once per epoch. The margin value $m$ for the triplet loss is 1. As mentioned, we use patches of $T = 1800$ frames for training and an initial auto-pool parameter $\alpha = 0$. If not already specified in Figure 4.1, the remaining hyperparameters and implementation details can be found at the project repository[33] (see Appendix C). We study the impact of the embedding dimension $d$ in the next section.

## 4.3 Results

### 4.3.1 Evaluation methods

For studying the effect of embedding dimension and performing the ablation study, we train MOVE with a subset of the training set, which includes 8,817 cliques and 44,909 tracks in total. The performance scores for those analyses are computed using the validation set. For comparison to the state of the art, we use the entire training set to train the model. We use the model weights obtained after the last epochs for all the results presented in this section. Below, we use MAP and MR1 scores to report performance (see Section 2.5.2).

For comparing the performance of MOVE with the state of the art, we use two datasets: the benchmark subset of Da-TACOS, and YouTubeCovers (YTC; see Section 2.5.1). To compare the performance on YTC with previous works, we follow their approach of only querying the test set to retrieve the versions in the reference set (Silva et al., 2016; Seetharaman & Rafii, 2017; Xu et al., 2018; Yu et al., 2019b).

[33] https://github.com/furkanyesiler/move

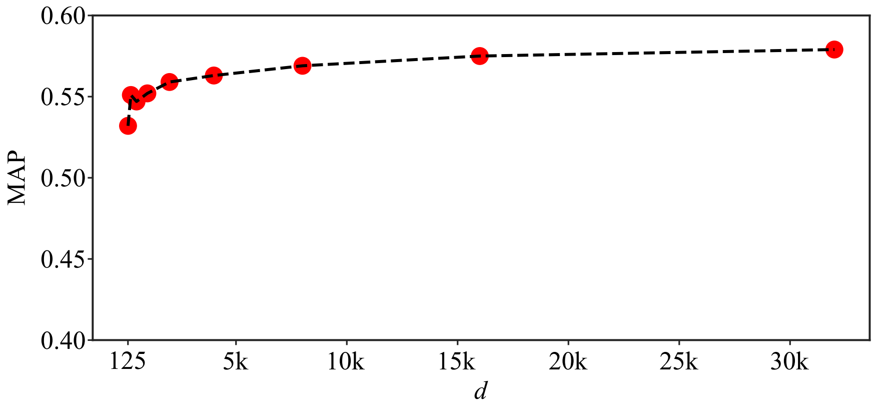


**Figure 4.5:** MAP with respect to embedding dimension $d$ on validation data.

Moreover, in this case, we remove from our training data the 17 cliques that overlap with YTC. After that, neither the benchmark subset of Da-TACOS nor YTC contains any overlapping cliques with respect to our training and validation data.

### 4.3.2 Effect of embedding dimension

For any embedding-based system, the size of the embeddings $d$ is a crucial hyper-parameter, as it can have an important effect on model performance. Therefore, we now study the model performance on the validation set with respect to it (Figure 4.5). For this set of experiments, we consider $d = \{128, 256, 512, 1\text{k}, 2\text{k}, 4\text{k}, 8\text{k}, 16\text{k}, 32\text{k}\}$. We observe that the performance continues to increase with the embedding dimensionality until it saturates at $d = 16\text{k}$. We can place a knee in the curve between $d = 512$ and $d = 2\text{k}$.

### 4.3.3 Ablation study

We then analyze the performance of the main components of MOVE by comparing them to their potential alternatives (Table 4.1). With that, we aim to quantify the importance of each decision. The first aspect we assess is the effect of the proposed data augmentation strategy (1). We find that removing data augmentation yields a relative decrease of 6% in MAP. The second aspect that we evaluate is the importance of the transposition invariance module explained in Section 4.2.2 (2). As an alternative, we consider the case where we do not preprocess the input by changing its shape and remove the max-pooling layer after the first convolution. Although trained with a much smaller learning rate ($10^{-4}$) and the Adam optimizer, the model was not able to properly learn an effective representation, even though multiple transpositions were present in the data augmentation function. The third aspect we consider is temporal summarization (3–6). We observe that the introduction of the auto-pool parameter $\alpha$ to multi-channel attention does not really change the results (3). In contrast, substituting the proposed multi-channel attention by auto-, max-, or average-pooling clearly

| | MAP | MR1 |
|---|---|---|
| MOVE | 0.575 | 156 |
| *Data augmentation* | | |
| 1: Without data augmentation | 0.540 | 180 |
| *Transposition invariance* | | |
| 2: Without transposition invariance | 0.154 | 399 |
| *Summarizing temporal content* | | |
| 3: Only multi-channel attention | 0.575 | 153 |
| 4: Only auto-pool | 0.563 | 145 |
| 5: Max-pooling | 0.561 | 152 |
| 6: Average-pooling | 0.491 | 197 |
| *Triplet mining strategies* | | |
| 7: Semi-hard mining | 0.545 | 135 |
| 8: Random mining | 0.427 | 167 |

**Table 4.1:** Ablation study. Performance on the validation set using $d = 16\,\text{k}$.

has a negative impact on performance (4–6). The final aspect we analyze is the effect of the triplet mining strategy (7–8). To do so, we train the network with online semi-hard (7) and random (8) mining strategies. For semi-hard mining, we pick a random positive example for each anchor and then select a negative example that satisfies the condition $E(\mathbf{v}_\text{A}, \mathbf{v}_\text{N}) \leq E(\mathbf{v}_\text{A}, \mathbf{v}_\text{P})$. In case no such negative example can be found, we pick a random one. For random mining, we randomly select one positive and one negative example for each anchor. We see that semi-hard and random mining result in a relative MAP decrease of 5 and 26%, respectively. Overall, our ablation study shows that all introduced variations have a positive impact on performance. The only exception is the mixing of the auto-pool parameter with multi-channel attention, which nonetheless does not substantially affect the performance.

### 4.3.4 Comparison with the state of the art

We now compare the performance of MOVE with the state of the art. The results on Da-TACOS (Table 4.2) show that MOVE clearly outperforms all the considered VI systems, excluding the ensembles. The relative MAP difference with respect to Qmax* (Serrà et al., 2009b), the most competing system, is over 13%. We also see that although the best performance is achieved with a relatively large embedding dimension of 16 k, a smaller embedding size of 4 k can still outperform the state of the art. When looking at the performance of ensemble systems, although MOVE yields a worse performance than the similarity network fusion (SNF) algorithm applied on Qmax and Dmax distances (Chen et al., 2018a), applying the same SNF algorithm on Qmax and MOVE distances result in a drastic increase in terms of MAP. Note that MR1 scores of both ensemble systems end up slightly higher than that of MOVE,

| | MAP | MR1 |
|---|---|---|
| *Individual systems* | | |
| 2DFTM (Bertin-Mahieux & Ellis, 2012) | 0.275 | 155 |
| SiMPle (Silva et al., 2016) | 0.332 | 142 |
| Dmax (Chen et al., 2018a) | 0.398 | 167 |
| Qmax (Serrà et al., 2009a) | 0.437 | 138 |
| Qmax* (Serrà et al., 2009b) | 0.445 | 130 |
| **MOVE** w/ $d = 4$k | **0.495** | **48** |
| **MOVE** w/ $d = 16$k | **0.507** | **46** |
| *Ensemble and fusion systems* | | |
| SNF (Qmax & Dmax) (Chen et al., 2018a) | 0.533 | 139 |
| SNF (Qmax & **MOVE** w/ $d = 16$k ) | 0.651 | 62 |

**Table 4.2:** Comparison of MOVE and other state-of-the-art VI systems on the Da-TACOS benchmark subset. Results for the proposed system are highlighted in bold.

which suggests that MOVE is better at retrieving versions at lower ranks compared with those ensemble systems.

We also evaluate MOVE on YTC (Table 4.3). The results support the claim that MOVE achieves state-of-the-art performance. However, we caution about the use of YTC to report VI performance, as differences measured with this dataset may not be significant due to the relatively small number of query and reference tracks (cf. Serrà, 2011). As an example, MOVE with $d = 4$k shows a similar result as the setting with $d = 16$k on YTC, while in larger datasets (i.e., the validation and benchmark subsets of Da-TACOS), the latter clearly outperforms the former.

### 4.3.5 Error analysis

Finally, we perform an error analysis to examine the failed cases and explore whether certain patterns exist in them. Since we use a relatively large dataset for the evaluation, our error analysis consists of randomly picked cases rather than an exhaustive inspection. We examine version and non-version pairs that yield high and low distance scores, respectively, using MOVE. We plot cremaPCP features for the selected pairs of failed cases to facilitate discovering any underlying patterns.

Figure 4.6 presents six pairs of tracks. The first three rows contain non-version pairs. Upon listening to the audio files, we hear that although having differences in rhythmic characteristics, the tracks in the second pair ("Poker Face" vs. "Need You Right Now") share the same chord progression almost throughout the entire duration. Therefore, having a low pairwise distance score using a model that processes only harmonic information is not unexpected. As for the first and the third pairs, we cannot hear notable similarities as their musical styles, rhythmic properties, and lyrics

| | MAP | MR1 |
|---|---|---|
| SiMPle (Silva et al., 2016) | 0.591 | 8 |
| 2DFTM sequences (Seetharaman & Rafii, 2017) | 0.648 | 8 |
| InNet (Xu et al., 2018) | 0.660 | 6 |
| SuCo-DTW (Silva et al., 2018) | 0.800 | 3 |
| CQT-TPPNet (Yu et al., 2019b) | 0.859 | 3 |
| **MOVE** w/ $d = 4$k | **0.889** | **3** |
| **MOVE** w/ $d = 16$k | **0.888** | **3** |

**Table 4.3:** Comparison of MOVE and other state-of-the-art VI systems on the YouTubeCovers dataset. Results for the proposed system are highlighted in bold.

are different. When we inspect Figure 4.6, we see resemblances in the cremaPCP representations for the first two pairs (i.e., a-b, and c-d). For the third pair, we see that cremaPCP representations are blurry, and we can confirm that the corresponding audio files contain high levels of noise that may obscure the efficient extraction of harmonic content.

The last three rows of Figure 4.6 contain version pairs to which MOVE assigns high distance scores. The styles of the songs are quite different for all the pairs; therefore, the results are not surprising. Apart from the differences in the harmonic content, the tracks for the second pair (versions of "You're the Cream in My Coffee") have less than optimal input representations, which may be the result of unusual equalization and "vinyl noise." When we check the cremaPCP representations for these pairs, we see that they also reflect important differences with respect to each other. Based on this analysis, we can conclude that using PCP representations as the only source of information in VI systems may fail in the following ways: (1) non-version pairs that have similar harmonic content can be mistaken as versions, (2) version pairs that have substantial changes in their harmonic characteristics can yield high distance scores, and (3) failing to extract useful harmonic information due to presence of noise or lack of prominent harmonic characteristics may fail the systems that depend on such input features.

## 4.4 Conclusion

In this chapter, we have proposed MOVE, a method for accurate and scalable version identification using musically motivated embeddings, which achieves state-of-the-art performance on two publicly available benchmark sets for VI. To make MOVE robust against common variations between versions, we have used explicit strategies for handling changes in pitch transposition and structure. Moreover, we have employed data augmentation functions that are particularly useful for VI. To train MOVE, we have used a metric learning approach by optimizing a triplet loss function. After

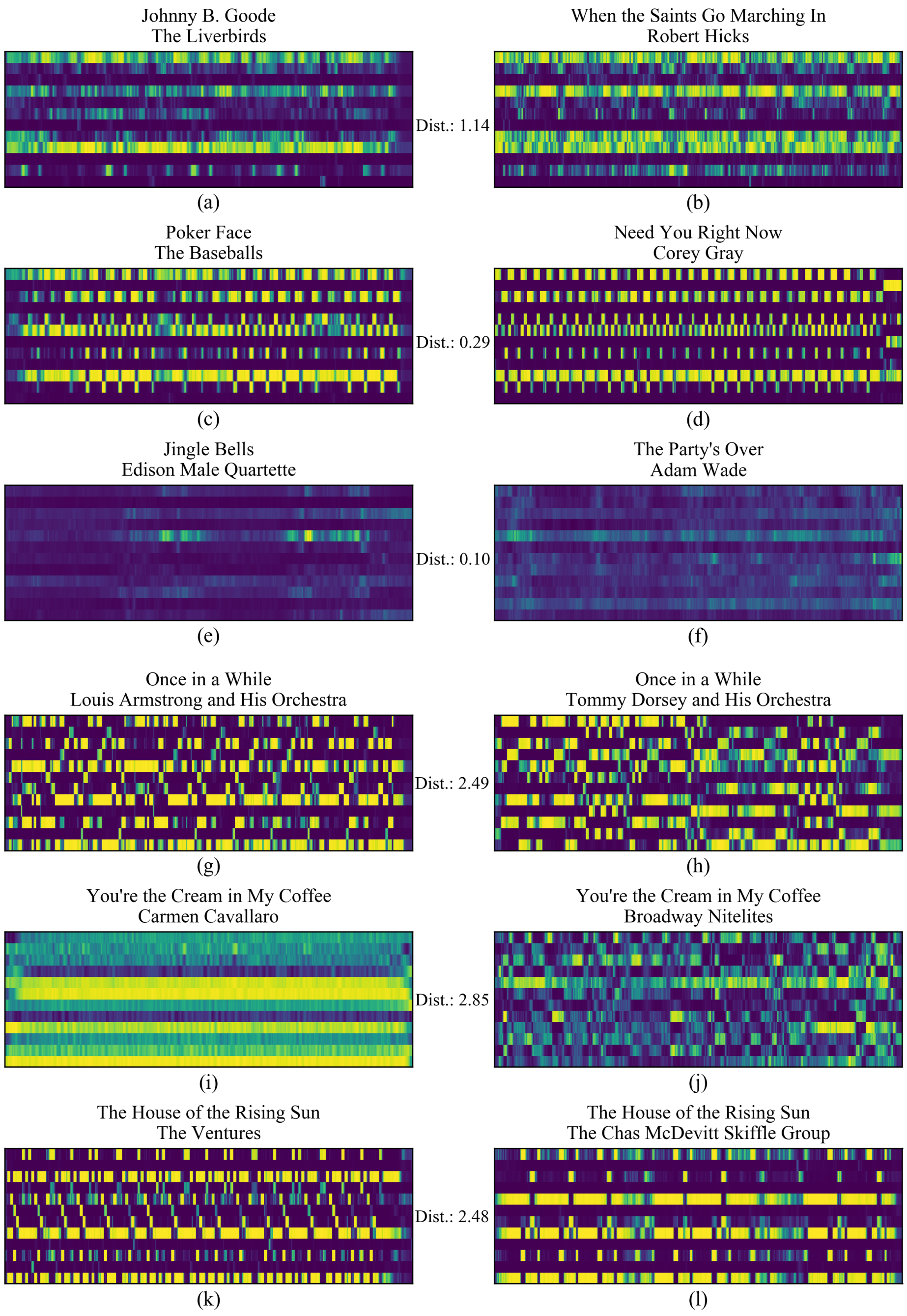


**Figure 4.6:** Randomly selected pairs of non-versions (the first three rows) and versions (the last three rows) for error analysis. The brighter colors indicate higher values. The cremaPCP features of each pair are transposed to maximize their similarities. The distance values indicated between pairs are obtained using MOVE.

motivating the components and training strategy of MOVE, we have performed an ablation study to justify our decisions. We have also studied the relationship between the embedding size and the performance of our model.

MOVE sets a strong baseline for data-driven VI research. Although it is considerably better at performance and computation times compared with its conventional alternatives, there are still certain limitations to further address: (1) MOVE processes only harmonic information from complex audio signals and, therefore, ignores an important portion of the potentially useful information, which is addressed in Chapter 5 by investigating ways to combine harmonic and melodic information in VI systems; (2) the size of the embeddings produced by MOVE is rather large compared to its alternatives, and this may result in increased storage size and longer retrieval times, which is addressed in Chapter 6 by exploring techniques to shrink the embeddings while maintaining the system performance; and (3) the network architecture, the training strategy, or the training data may cause MOVE to favor certain groups of artists over others, which is addressed in Chapter 7 by creating a framework to assess the algorithmic bias in VI systems.

# Chapter 5

# Improving Accuracy with Data-driven Fusion

## 5.1 Introduction

PCP variants have long been the most used features in VI (see Section 2.3.1), and based on their previous success, we have decided to employ them as the input representation for MOVE. However, considering the complexity of musical audio signals, using only a single source of information (i.e., harmonic) for identifying links between version pairs is surely a suboptimal strategy. A number of possible issues that may arise from using only harmonic features were outlined by Serrà (2011).

Contrarily to VI systems, which exploit harmonic features better than other sources of musical information, humans can identify version pairs mostly using melodic or textual information (e.g., lyrics). Nevertheless, difficulties in obtaining and processing such information (e.g., extracting melody or lyrics from polyphonic audio signals) kept researchers from fully exploiting them in audio-based VI. While obtaining lyrics from musical audio is still a significant challenge, there has been major progress in melody estimation, which has also adopted data-driven solutions like VI.

On another note, before data-driven approaches became dominant, VI systems were having difficulties in exceeding the accuracy scores obtained with local alignment–based systems that used PCP variants as features. Although considerable progress was made in terms of developing faster systems, developing more accurate systems was problematic. To overcome this challenge, researchers investigated ways to combine information obtained from separate systems, namely, feature fusion (see Section 2.4.2) and ensemble methods (see Section 2.4.3).

On the one hand, feature fusion approaches investigated ways of combining information at different stages of the VI workflow (i.e., early or late fusion) for a single item without considering relationships between items. On the other hand, ensemble approaches were designed to combine information from multiple systems after they estimate separate similarity scores between pairs of items, to obtain a refined similarity score. Both feature fusion and ensemble methods showed promising results for accuracy improvements; however, they were struggling with the accuracy–scalability trade-off, which made such systems almost impractical to use in large-scale use cases.

Recent years have provided researchers access to well-performing melody estimation algorithms and both fast and accurate VI systems. Therefore, we believe that it is a good time to reinvestigate feature fusion and ensemble methods to combine systems that process melodic and harmonic information extracted from musical audio signals. In this chapter, we present our work toward this goal. For this, we select four input representations (a dominant melody, a multi-pitch, and two harmonic features) and design a novel deep learning–based VI system that can work with all the selected features. Firstly, we train different instances of the same model with each feature to determine their individual performances and to create baselines. Secondly, after obtaining the baselines, we experiment with a simple distance averaging scheme to combine pairwise distances obtained from different models (i.e., an ensemble approach). Finally, we propose a data-driven fusion scheme to let the individual models learn a better way to combine the complementary information they carry for each item (i.e., a feature fusion approach). Both the simple distance averaging and data-driven schemes obtain state-of-the-art results on two publicly available datasets, with drastic improvements in system accuracy.

This chapter is based on Doras et al. (2020).

## 5.2 Methods

The main goal of this study is to investigate whether combining information from systems that process different musical characteristics can increase the overall accuracy. Since we aim to observe a performance improvement with respect to using individual systems, our first task is to evaluate each of such systems to set a baseline. For this, we train separate, randomly initialized instances of the same VI model with each feature alone. Although MOVE yields state-of-the-art performance, its initial layers are specialized for processing PCP features. Therefore, we design a new VI model that incorporates more generic layers to allow the processing of input features of different characteristics and shapes. After training and evaluating this new model using each input feature alone (i.e., creating the baselines), we design an ensemble and a feature fusion scheme to test our hypothesis that different musical characteristics carry complementary information and combining such information may boost overall performance.

### 5.2.1 Input representations

We consider a total of four features for this study: plain PCP (pPCP), cremaPCP, dominant melody (dMel), and multi-pitch (mPitch). While the first two represent

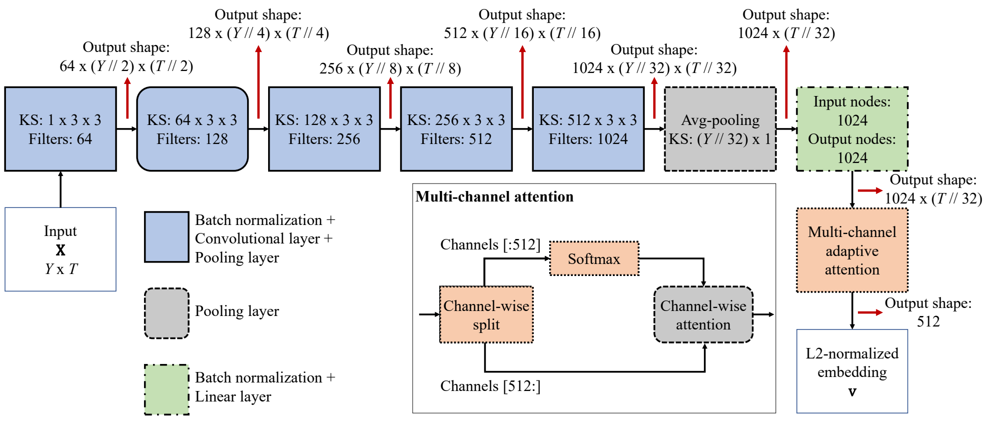

Output shape: 64 x (Y // 2) x (T // 2)
Output shape: 128 x (Y // 4) x (T // 4)
Output shape: 256 x (Y // 8) x (T // 8)
Output shape: 512 x (Y // 16) x (T // 16)
Output shape: 1024 x (Y // 32) x (T // 32)
Output shape: 1024 x (T // 32)
KS: 1 x 3 x 3 Filters: 64
KS: 64 x 3 x 3 Filters: 128
KS: 128 x 3 x 3 Filters: 256
KS: 256 x 3 x 3 Filters: 512
KS: 512 x 3 x 3 Filters: 1024
Avg-pooling KS: (Y // 32) x 1
Input nodes: 1024 Output nodes: 1024
Output shape: 1024 x (T // 32)
Input X Y x T
Batch normalization + Convolutional layer + Pooling layer
Pooling layer
Batch normalization + Linear layer
Multi-channel attention
Channels [:512]
Softmax
Channel-wise split
Channel-wise attention
Channels [512:]
Multi-channel adaptive attention
Output shape: 512
L2-normalized embedding v

**Figure 5.1:** The network architecture of MICE. KS denotes kernel size, and // denotes the integer divide operation.

harmonic information in the musical audio, the third one represents the melodic information, and the last one contains both melodic and harmonic information (see Section 2.3.1). All the features are extracted using a hop length of about 93 ms. As done for MOVE, both PCP features are preprocessed to transform their shapes in the pitch class dimension, from $\mathbf{X} \in [0,1]^{12\times T}$ to $\hat{\mathbf{X}} \in [0,1]^{23\times T}$, so that they contain all possible transpositions of the original input (see Section 2.4.1). Both dMel and mPitch are originally computed with a resolution of five bins per semitone across six octaves (i.e., 360 bins), and we downsample them by a factor of 5 via linear interpolation. For dMel, only the three octaves around the mean pitch are considered for each track since it is a quite sparse representation. In summary, the shapes of the features in the y-axis are 23, 23, 36, and 72 for pPCP, cremaPCP, dMel, and mPitch, respectively. Finally, each input representation is normalized to have values between 0 and 1.

### 5.2.2 Network architecture

MOVE, as explained in Chapter 4, shows state-of-the-art performance thanks to the musically motivated design decisions it incorporates. However, the transposition invariance module that it employs is specifically designed for processing PCP features. Since our goal for the present study is to learn and combine models that process different input features, we have to design a more generic network architecture that can handle features other than PCPs as well. For this, we use the model proposed by Doras & Peeters (2019) as a foundation and improve it using a few of the modules and training strategies of MOVE. We call the resulting model MICE: musically informed cover embeddings (see Figure 5.1).

MICE consists of five convolutional blocks, each of which includes a batch normalization layer, a convolutional layer with kernel shape $3 \times 3$, and an average-pooling layer with kernel and stride of $2 \times 2$. The number of convolution kernels is doubled at each level: the first block has 64 layers while the last one has 1024. The output of the last convolutional block is averaged along the frequency axis, and a linear layer is applied. After obtaining a representation with shape $1024 \times T'$, where $T'$ is the number of time frames, MICE uses the temporal aggregation module of MOVE (see Section 4.2.2), to convert the representation into a vector of size 512, regardless of $T'$. The resulting vector is then L2-normalized and considered as the embedding vector for the input track.

### 5.2.3 Comparing features

To set baselines for the ensemble and fusion experiments, we first train four separate instances of MICE using each of the considered input representations. Since MICE is designed to handle inputs of different shapes, no additional changes are made to accommodate different features. To minimize the differences between the training processes, the details described below are kept the same for all four models.

**Objective function —** We use a triplet loss function (see Equation 4.3) to train MICE. However, there are two main differences compared with the process used for training MOVE. Firstly, although we use the squared Euclidean distance function to compute distances between embeddings, we do not normalize the distance scores by the embedding dimension. The reason for this is that since the embeddings are L2-normalized, the distance scores are bounded between [0, 4] regardless of the embedding size. Secondly, we use a slightly different online hard negative mining scheme to select the triplets involved in the training process, in contrast to the online hard triplet mining scheme used for training MOVE. For this, each item in a mini-batch is considered as an anchor (A) and each of its versions in the mini-batch is considered as a positive (P) example (as opposed to selecting the hardest positive example). As the negative (N) example for each of the (A, P) pairs, we select the one that yields the maximum distance with respect to the anchor while satisfying the condition, $||\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{N}||^2 < ||\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{P}||^2$, where $||\ ||$ corresponds to the Euclidean norm and $\mathbf{v}_\mathrm{A}$, $\mathbf{v}_\mathrm{P}$, and $\mathbf{v}_\mathrm{N}$ denote the embedding vectors of A, P, and N, respectively (as opposed to selecting the hardest negative example). If there are no such N, we select the one with the lowest distance with respect to the anchor.

**Training data —** To train instances of MICE, we use the publicly available SHS5+ dataset (see Section 2.5.1), which includes +60 k tracks split into 7.5 k cliques. SHS5+ includes all the input features considered in this work in the pre-extracted form. We create training and validation partitions from SHS5+ using a ratio of 4 to 1 with respect to the number of cliques.

**Hyperparameters and optimization —** We train each model using the Adam optimizer, with an initial learning rate of $10^{-4}$. The learning rate is reduced by a factor of 2 each time the validation loss stops decreasing for 5 k iterations. The models are trained for 50 k iterations, or until the learning rate falls below $10^{-7}$. The size of the mini-batches is 64. The margin hyperparameter $m$ in the triplet loss is set to 1.

### 5.2.4 Combining features

After obtaining our baselines, we start to investigate possible ensemble and fusion schemes to combine information from models trained with different input features. Below, we explain two main ideas that we explore in this work.

#### 5.2.4.1 Distance averaging

The ensemble scheme we use is a simple distance averaging operation. Given two instances of MICE models trained with different input features (i.e., $C^{\text{feat1}}$ and $C^{\text{feat2}}$), a pair of items (i.e., $i$ and $j$), and a pair of input features for each item (i.e., $\mathbf{X}_i^{\text{feat1}}$, $\mathbf{X}_i^{\text{feat2}}$, $\mathbf{X}_j^{\text{feat1}}$, and $\mathbf{X}_j^{\text{feat2}}$), the performed operation can be represented in mathematical notation as,

$$E_{i,j}^{\text{DA}} = \frac{1}{2}(\|C^{\text{feat1}}(\mathbf{X}_i^{\text{feat1}}) - C^{\text{feat1}}(\mathbf{X}_j^{\text{feat1}})\|^2 + \|C^{\text{feat2}}(\mathbf{X}_i^{\text{feat2}}) - C^{\text{feat2}}(\mathbf{X}_j^{\text{feat2}})\|^2). \quad (5.1)$$

We hypothesize that even such a simple averaging scheme can allow a model that is confident (i.e., producing a very high or low distance) to affect the final decision in a favorable way. The fact that the models are trained with features representing different musical characteristics increases the chances of having at least one model producing confident results. Moreover, this ensemble scheme does not require any retraining or fine-tuning of the models, and we do not need to normalize the distance scores before averaging them since the range of distances is bounded due to L2-normalization applied to the embeddings.

#### 5.2.4.2 Data-driven late fusion

Our fusion idea is to learn a new embedding using the individual embeddings from different models. Although requiring a training process, which is not the case for the distance averaging scheme, this idea addresses some of the drawbacks of the first approach. Firstly, distance averaging is likely a suboptimal strategy to combine information since it is a general approach that does not consider the characteristics of the sources. A data-driven approach, on the other hand, optimizes the final embeddings in a way that the contribution of the individual embeddings is determined using the training data. Secondly, the distance averaging scheme requires having multiple embeddings for the same track and, thus, multiple comparisons for the same pair of tracks[34]. Indexing several vectors and combining multiple results may complicate

[34]Note that this issue can be avoided by storing the concatenated embeddings per track and using the squared Euclidean distance function as the squared Euclidean distance between the concatenated embeddings corresponds to the sum of the squared Euclidean distances between individual embeddings for two a pair of tracks. However, this practical consideration does not easily extend to some other distance functions like the cosine distance.

the operational usage of the system. By having a single embedding, the data-driven approach alleviates this complexity.

Given two models $C^{\text{feat1}}$ and $C^{\text{feat2}}$, and two input representations $\mathbf{X}_i^{\text{feat1}}$ and $\mathbf{X}_i^{\text{feat2}}$ for the item $i$, this scheme can be represented as follows:

$$\hat{\mathbf{v}}_i = L([C^{\text{feat1}}(\mathbf{X}_i^{\text{feat1}}), C^{\text{feat2}}(\mathbf{X}_i^{\text{feat2}})]), \tag{5.2}$$

where $\hat{\mathbf{v}}_i$ is the final embedding, $L$ is the late fusion model to be trained, [.] is the concatenation operation. The models $C^{\text{feat1}}$ and $C^{\text{feat2}}$ can be pretrained, or randomly initialized and trained from scratch along with $L$.

For the experiments that use cremaPCP features, we use MOVE with the embedding size $d = 512$. For all the other features, we use MICE. As the late fusion model, we consider only a single linear layer without any activation function. The final embeddings are L2-normalized.

We consider three configurations for this set of experiments: (1) each branch model (i.e., $C^{\text{feat1}}$ and $C^{\text{feat2}}$) is randomly initialized and trained from scratch along with the late fusion model (LF-a), (2) each branch model is pretrained separately and fine-tuned while the late fusion model is being trained (LF-b), and (3) each branch is pretrained but their weights are frozen during the training of the late fusion model (LF-c).

**Objective function —** We use a triplet loss function to train (or fine-tune) the considered models. All the training settings including the triplet mining strategies are the same as explained in Section 5.2.3.

**Training data —** To test the robustness of the considered fusion schemes, we train the models separately with both SHS5+ and the Da-TACOS training subset (see Section 3.2).

**Hyperparameters and optimization —** The training procedure is the same as described in Section 5.2.3, except that the learning rates for LF-b and LF-c are initialized at $5 \times 10^{-6}$ and $10^{-1}$, respectively.

## 5.3 Results

### 5.3.1 Evaluation methodology

To evaluate models trained with the Da-TACOS training subset and SHS5+, we use the Da-TACOS benchmark subset and SHS4-, respectively (see Section 2.5.1). To report the model performances, we use MAP and MR1 metrics (see Section 2.5.2).

### 5.3.2 Comparing individual systems

Our first set of experiments aim to compare the performances of four instances of MICE, each trained with a different input feature. Table 5.1 presents the results of

| Input | MAP | MR1 |
|---|---|---|
| dMel | 0.412 | 1431 |
| mPitch | 0.422 | **862** |
| pPCP | 0.174 | 1465 |
| cremaPCP | **0.499** | 1169 |

**Table 5.1:** Results on SHS4- for considered input features.

| Input | MAP | MR1 |
|---|---|---|
| dMel+mPitch | 0.571 | 614 |
| dMel+cremaPCP | **0.679** | **529** |
| mPitch+cremaPCP | 0.627 | 593 |
| dMel+cremaPCP (O) | 0.873 | 115 |

**Table 5.2:** Comparison of ensemble systems with various input feature combinations using the distance averaging scheme on SHS4-. O denotes “oracle” (see Section 5.3.3.1).

this set of experiments. The models trained with cremaPCP and mPitch outperform their alternatives in terms of MAP and MR1, respectively. The performance difference between cremaPCP and pPCP confirms the findings from our initial experiments presented in Section 3.5. The fact that cremaPCP and mPitch outperform dMel highlights the importance of harmonic information for VI. In conclusion, the results suggest that cremaPCP offers both higher performance and lower memory footprint compared to its alternatives.

## 5.3.3 Ensemble and fusion experiments

### 5.3.3.1 Distance averaging

Table 5.2 presents the results of the experiments for the distance averaging scheme. We consider all the combinations excluding the ones with pPCP due to its low performance in the previous set of experiments. First of all, we see that all the combinations yield higher performance scores than the baselines, which confirms our hypothesis on increasing accuracy by combining multiple sources of information. Secondly, the combinations with cremaPCP yield notably higher scores than their alternative. However, although cremaPCP and mPitch are the highest performing features in the baseline (see Table 5.1), their combination is outperformed by the combination of cremaPCP and dMel. This suggests that the information incorporated in cremaPCP and dMel features are more complementary to each other than that of other combinations.

To develop a better understanding of the results, Figure 5.2 illustrates the normalized distances of randomly selected version and non-version pairs, obtained from MICE models trained with cremaPCP, mPitch, and dMel features. Overall, all the models

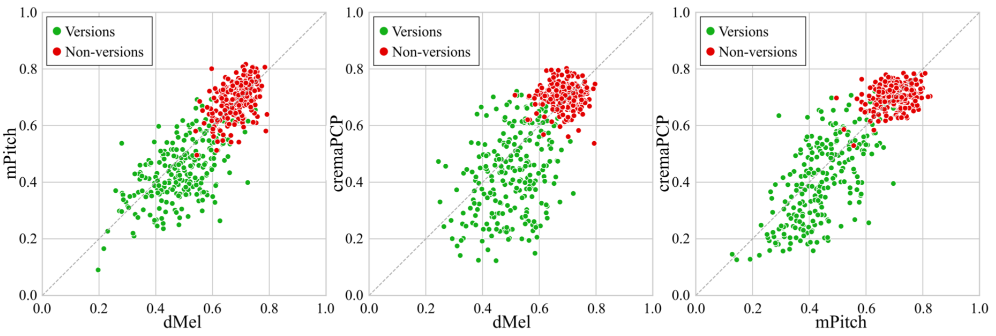


**Figure 5.2:** Comparison of the normalized distances computed for the pairs from SHS4- (version pairs in green and non-version pairs in red) with different features: dMel vs. mPitch (left), dMel vs. cremaPCP (middle), mPitch vs. cremaPCP (right). For clarity, only 500 randomly picked pairs are plotted (250 versions and 250 non-versions).

yield normalized distance scores between 0.6 and 0.8 for non-version pairs. However, for version pairs, the points are more scattered in the middle figure that shows the cremaPCP and dMel distances, which supports the empirical results on their complementarity.

Lastly, the bottom row of Table 5.2 shows the performance of an "oracle" for the cremaPCP and dMel combination, which chooses the lowest or highest distance obtained from the source models for version or non-version pairs, respectively. The performance gap between the distance averaging results and the oracle confirms our hypothesis that this scheme is suboptimal for combining information from multiple sources.

#### 5.3.3.2 Data-driven late fusion

The results of the data-driven fusion experiments can be seen in Table 5.3. For this set of experiments, we consider only the combination of cremaPCP and dMel since it outperforms its alternatives for the distance averaging scheme. Note that we use MOVE to process cremaPCP features instead of MICE.

Firstly, all the three late fusion schemes outperform the source models by large margins, which confirms that the models learn efficient ways of combining information from their sources. Secondly, LF-a yields a worse performance than its alternatives, which suggests that training the source models simultaneously with the late fusion model hinders their effectiveness. Thirdly, while LF-b and LF-c yield similar performances on SHS4-, the gap in MAP is larger on Da-TACOS. This suggests that training only the final fusion layer (LF-c) is more effective than doing the same while fine-tuning the source models (LF-b). Lastly, the distance averaging scheme outperforms all the data-driven fusion schemes on SHS4- while it is outperformed by LF-c on Da-TACOS. This emphasizes the importance of evaluating VI models on multiple datasets.

| | Da-TACOS | | SHS4- | |
|---|---|---|---|---|
| Input | MAP | MR1 | MAP | MR1 |
| dMel (MICE) | 0.360 | 94 | 0.412 | 1431 |
| cremaPCP (MOVE) | 0.484 | 59 | 0.533 | 1188 |
| dMel+cremaPCP (Avg.) | 0.621 | 32 | **0.697** | **517** |
| dMel+cremaPCP (LF-a) | 0.570 | **29** | 0.617 | 686 |
| dMel+cremaPCP (LF-b) | 0.592 | 32 | 0.655 | 655 |
| dMel+cremaPCP (LF-c) | **0.635** | 30 | 0.660 | 657 |

**Table 5.3:** Comparison of the considered ensemble and fusion schemes on the Da-TACOS benchmark set (left) and SHS4- (right). Avg. and LF denote distance averaging and late fusion, respectively. Note that cremaPCP and dMel+cremaPCP (Avg.) scores are higher here than in Table 5.2 because cremaPCP is now processed by MOVE.

| | MAP | MR1 |
|---|---|---|
| *Individual systems* | | |
| 2DFTM (Bertin-Mahieux & Ellis, 2012) | 0.275 | 155 |
| SiMPle (Silva et al., 2016) | 0.332 | 142 |
| Dmax (Chen et al., 2018a) | 0.398 | 167 |
| Qmax (Serrà et al., 2009a) | 0.437 | 138 |
| Qmax* (Serrà et al., 2009b) | 0.445 | 130 |
| MOVE-512 | 0.484 | 59 |
| MOVE-16 k | 0.507 | 46 |
| *Ensemble and fusion systems* | | |
| SNF (Qmax & Dmax) (Chen et al., 2018a) | 0.533 | 139 |
| SNF (Qmax & MOVE-16 k) | 0.651 | 62 |
| **Avg. (MOVE-512 & MICE w/ dMel)** | **0.621** | **32** |
| **LF-c (MOVE-512 & MICE w/ dMel)** | **0.635** | **30** |

**Table 5.4:** Comparison of the proposed approaches and other state-of-the-art VI systems on the Da-TACOS benchmark subset. MOVE-512 and MOVE-16 k denote MOVE models that produce embeddings of size 512 and 16 k, respectively. Results for the proposed systems are highlighted in bold.

### 5.3.4 Comparison with the state of the art

We now compare the performance of our ensemble and fusion systems to the state of the art on the Da-TACOS benchmark set (Table 5.4). Both the distance averaging and data-driven fusion schemes outperform all the individual systems by considerable margins in both MAP and MR1. Compared with other ensemble systems, although some similarity network fusion (SNF) methods output higher MAP scores, the approaches proposed in this chapter have a clear advantage in terms of MR1, which suggests that the proposed approaches can retrieve a relevant result at lower ranks than the alternatives. Moreover, SNF approaches are not suitable for cases where only a single item is queried against a reference corpus because the SNF algorithm requires fully connected graphs to process. Therefore, from a practical point of view, our proposed solutions may be considered to provide a better alternative.

### 5.3.5 Error analysis

Lastly, we perform an error analysis to explore the cases where two models that use different input representations result in contradicting decisions. For this, we select two version pairs and two non-version pairs, and we plot the dominant melody and cremaPCP features for each track. Although listening to the tracks can reveal important information for us to gain insight on possible reasons for the failed cases, our models use such feature representations as input; therefore, inspecting those features may facilitate our analysis.

Figure 5.3 presents the dominant melody and cremaPCP features for four pairs of tracks. We see that for the first version pair, the cremaPCP features (subfigures c and d) contain visible differences, which suggests differences in their harmonic characteristics. Their dominant melody features (subfigures a and b), on the other hand, demonstrate similar contours, which supports the low distance score obtained by using a model that processes melodic information. For the second version pair, the situation is the opposite. While the cremaPCP features clearly resemble each other (subfigures g and h), their melody features reflect more differences with respect to each other. Therefore, it is a good idea to use melodic information for the first version pair while harmonic information is more useful for identifying the second pair.

The last four rows of Figure 5.3 contains the same features for two non-version pairs. We see that using cremaPCP features (subfigures k and l) may lead us to identify this pair of tracks as versions; however, their melody features contain substantial differences for us to reach the conclusion that these tracks are indeed not versions of each other. For the last pair of tracks, we see that cremaPCP features result in a more confident decision that these tracks are not versions compared to melody features. The aforementioned observations on both version and non-version pairs suggest that using a single musical characteristic (e.g., harmonic or melodic) for identifying versions is certainly suboptimal due to the complexity of musical tracks and their relationships.

## 5.4 Conclusion

In this chapter, we have explored ensemble and fusion methods that combine information from individual systems processing certain types of musical characteristics. In particular, we have experimented with systems that use PCP, dominant melody, and multi-pitch features as input. After setting baselines using different instances of the same deep learning–based model, each trained with a different feature, we have evalu-

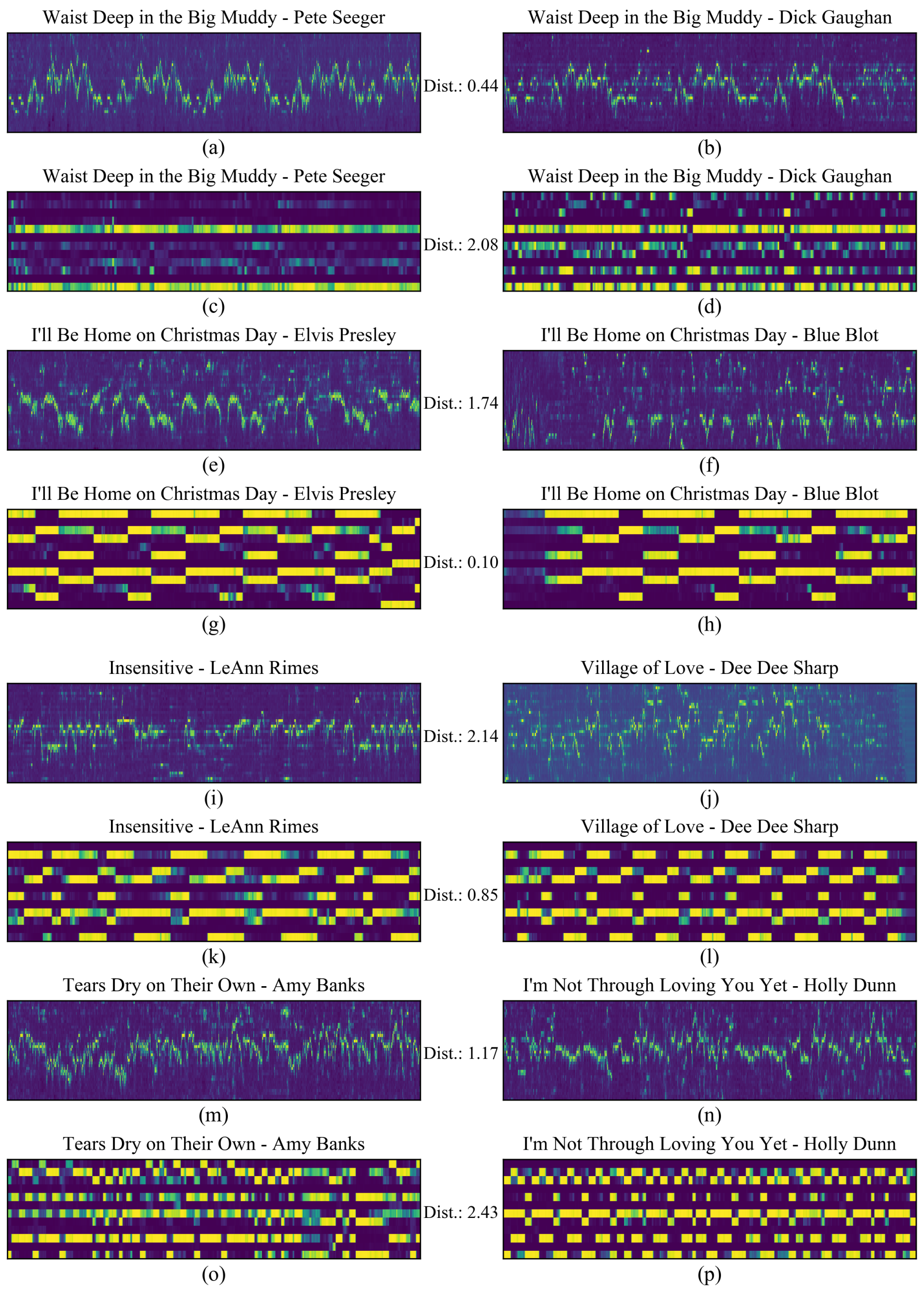


**Figure 5.3:** Dominant melody and cremaPCP features for two pairs of versions (a–h) and two pairs of non-versions (i–p). The brighter colors indicate higher values. Logarithm of the dominant melody features are taken for better illustrations. The cremaPCP features of each pair are transposed to maximize their similarities. The distances between subfigures are computed using MICE with dominant melody or cremaPCP inputs.

ated an ensemble and a feature fusion approach. For the ensemble approach, we have aggregated pairwise distances obtained from two models trained with two different features, which has yielded a substantial performance gain compared with the individual models that form the ensemble. For the fusion approach, we have developed a data-driven fusion system to obtain further improvements. Using two models that process harmonic and melodic features, we have obtained two embeddings for each track. We have then trained a linear layer to learn how to combine those embeddings into a single one. The fusion approach has also resulted in drastic improvements in performance, comparable to the distance aggregation scheme. We have evaluated the proposed methods using two publicly available datasets and demonstrated state-of-the-art performance.

The starting point for the experiments described in this chapter has been the fact that MOVE (see Chapter 4) processes only harmonic information from the musical audio signals. Through the experiments and the error analysis performed in this chapter, we have shown the benefits of using multiple musical characteristics for a better performance in VI. However, such accuracy improvements come at the expense of longer computation times, and providing fast solutions is a crucial requirement for many industrial use cases. In the next chapter, we address the scalability aspect of VI systems and investigate techniques to reduce the size of the embeddings obtained from neural network–based VI models for speeding up the retrieval process.

# Chapter 6

# Improving Scalability with Embedding Distillation

## 6.1 Introduction

Building systems that are both accurate and scalable is one of the main goals of this dissertation. The reason for this is primarily related to the VI use cases in commercial settings, an important one of which is to detect copyright infringement cases in media streaming platforms and live performance events. Such application scenarios require having fast and reliable solutions. For example, more than 500 h of video content are uploaded to YouTube every minute[35], and handling the music licensing aspect of them (i.e., identifying the cases where a video includes a copyrighted piece of music) requires having systems that can process a vast amount of data and still yield sufficient performance.

We have mentioned in Section 2.2 that the early, alignment-based systems incorporated musical know-how to capture similarities between versions, resulting in strong performances but long computation times. With the release of the Million Song Dataset (Bertin-Mahieux et al., 2011), researchers were further encouraged to address the scalability issue by exploring embedding-based systems that encode tracks into more compact vectors. Although offering significant improvements for scalability, the performance of such systems failed to match their predecessors (see Section 3.4). The pioneering deep learning–based systems, on the other hand, showed substantial progress in both accuracy and scalability for VI (Doras & Peeters, 2019; Yu et al., 2019b; Zalkow & Müller, 2020), but they were only evaluated in academic settings where the size of the evaluation datasets was, at most, in the magnitude of tens of thousands. With catalogs of such size, neither the required space for storing the embeddings nor the required computation time for the retrieval process is critical. However, when

[35]https://www.cnbc.com/2018/03/14/with-over-1-billion-users-heres-how-youtube-is-keeping-pace-with-change.html

working with catalogs of millions of tracks, as in industrial applications, the size of the embeddings in which the individual tracks are encoded may have a crucial impact on both the storage space and the retrieval times. Therefore, developing specific methodologies for obtaining smaller embedding vectors without sacrificing the system performance is, and further will be, important for deploying VI systems for industrial applications.

In this chapter, we present our investigations toward improving the scalability of existing embedding-based VI systems that use neural networks as encoders. Specifically, the goal of this study is to reduce the size of embedding vectors without compromising the accuracy of the systems. Since embeddings can be precomputed, reducing their size is crucial to improve data storage and, more importantly, retrieval time. For this purpose, we consider three core state-of-the-art strategies, namely, unsupervised dimensionality reduction, neural network pruning, and knowledge distillation. Apart from introducing a number of techniques from other fields to VI research, we also consider a novel knowledge distillation loss for metric learning, which aims to optimize a clustering evaluation metric. Moreover, inspired by transfer learning applications, we propose a technique called latent space reconfiguration, to show that learning a compact and efficient latent space is facilitated by using a pretrained feature extractor due to its stronger priors, compared with using a randomly initialized one. The experiments reported here show that the performance of a pretrained network can be preserved or even improved while shrinking the embedding vectors down to less than 1% of their original sizes. We evaluate our approach on a publicly available test set and share our code, instructions for using a newly contributed training dataset, and supplementary materials on the project repository[36] (see Appendix C).

This chapter is based on Yesiler et al. (2020b).

## 6.2 Methods

In this study, we focus on a set of techniques for improving the scalability of existing VI systems in the retrieval phase by reducing the size of the embeddings, rather than building a novel network architecture.[37] We hypothesize that a high-capacity encoder (i.e., the base model) is needed to extract the essential information from complex and noisy signals such as current tonal representations. However, once a reliable encoder is obtained, it can be used for training a second model (i.e., the reduced model) that outputs embeddings with a lower dimensionality, ideally without compromising

[36]https://github.com/furkanyesiler/re-move

[37]To illustrate the benefits of using smaller embeddings, consider computing distances between a query and a reference database with 10 M embeddings. This takes us (with a simple brute-force, double-loop Euclidean distance function) 0.32 s using $d = 256$, but the elapsed time increases up to 4.75 s for $d = 4k$, and to 18.43 s for $d = 16k$ (the embedding size of MOVE; see Chapter 4). Although the absolute values are subject to change based on computational resources, for industrial applications on portable devices, such differences in magnitude for the retrieval time (from 0.32 to 18.43 s) may drastically affect user experience and product appeal.

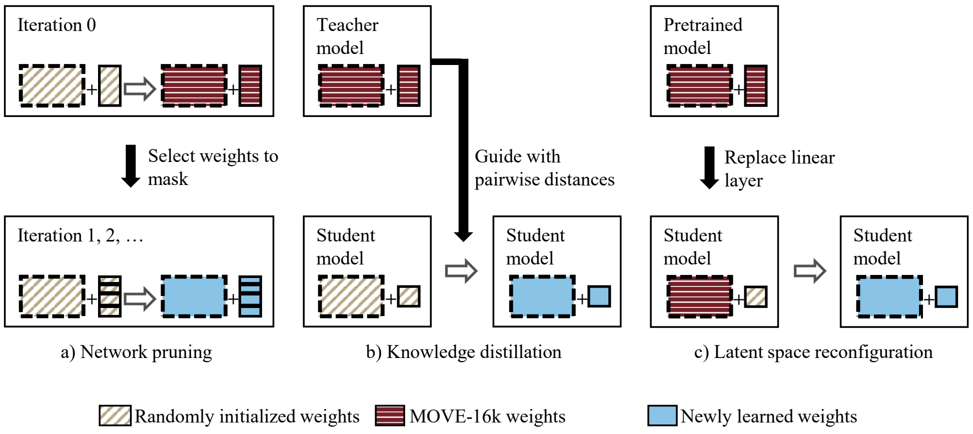


**Figure 6.1:** An overview of neural network–based embedding distillation methods. The hollow arrows denote training process, the boxes with dashed and with solid outlines denote feature extractors and fully connected layers, respectively.

accuracy. Due to the goal of using large embeddings to have smaller ones that yield similar performances, we call this set of methods "embedding distillation" techniques.

### 6.2.1 The base model

Our methods require starting from a pretrained and sufficiently reliable model. For this, we use MOVE (see Chapter 4), together with its pretrained weights, which are available publicly. Nonetheless, we believe that all the methods introduced in this section can be applied to other embedding-based systems using neural networks (initial results are available in the project repository; see Appendix C).

MOVE outputs embedding vectors $\mathbf{v} = F(\mathbf{X}) \in \mathbb{R}^d$, where $F$ is the MOVE model, $\mathbf{X}$ is the input feature, and $d$ is the embedding size. We have seen in Section 4.3.2 that the MAP scores increase with the size of the embeddings. Figure 4.5 shows results for $d$ between 128 and 32 k where a clear accuracy drop happens for $d < 2048$ (for the final model, we have chosen a rather high dimensionality $d = 16$ k). In contrast, the dimensionalities we consider here for the reduced model are $d = \{128, 256, 512, 2048\}$.

### 6.2.2 Embedding distillation techniques

We now explain the embedding distillation techniques we consider in this study. We experiment with a set of existing techniques that include classical dimensionality reduction, neural network pruning, and knowledge distillation. We also propose a knowledge distillation scheme based on internal cluster evaluation metrics and another method that we call latent space reconfiguration, which is inspired by transfer learning schemes. Figure 6.1 illustrates an overview of the neural network–based embedding distillation techniques.

#### 6.2.2.1 Classical unsupervised techniques

Before going into complex solutions, we investigate the benefits of using classical dimensionality reduction techniques for embedding distillation. For this, we use principal component analysis (PCA), independent component analysis (ICA), and Gaussian random projection (GRP) techniques. Each model is fit using the training set embeddings obtained with MOVE and applied to the evaluation set embeddings. We use the implementations from the scikit-learn library (Pedregosa et al., 2011) and change only the number of target components.

#### 6.2.2.2 Pruning

Pruning a large neural network can preserve the original performance while eliminating more than 90% of its weights (LeCun et al., 1989; Hanson & Pratt, 1988; Han et al., 2015; Frankle & Carbin, 2019). The main challenge is to identify the importance of weights and connections. Previous research explored the use of absolute weight magnitudes (Hanson & Pratt, 1988; Han et al., 2015; Frankle & Carbin, 2019) and the Hessian of the loss function (LeCun et al., 1989). Pruning operations can be performed layer- or network-wise, in a one-shot or an iterative fashion, and combined with quantization or clustering. To the best of our knowledge, at the time of this study, network pruning had not been considered for VI, nor further explored in MIR systems in general.

Based on the approach of Han et al. (2015), we investigate whether we can prune the dimensions of the latent space constructed by MOVE in an iterative way. Although pruning the weights of all layers throughout the network is the most common practice, the underlying idea can be applied to only the final linear layer of the model in order to obtain embeddings with fewer dimensions. Denoting the weights of the final linear layer of MOVE as $\mathbf{W} \in \mathbb{R}^{d\times 256}$, where $d$ is the size of the embeddings and 256 is the number of input connections to the linear layer, our method computes the mean of the absolute values per row for $\mathbf{W}$ and sorts the rows based on these mean values. At the end of each iteration $b$ ($b \in \{0,1,...\}$), the weights of the top 50% rows are restored to their initial values from iteration 0 and retrained. The weights of the bottom 50% rows are zeroed-out and not considered for the next iterations (i.e., they are "pruned"; see Figure 6.1(a)).

#### 6.2.2.3 Knowledge distillation

Bucilă et al. (2006), and later Hinton et al. (2015), explored the idea where a small neural network (i.e., a student model) is trained with the guidance from a wide, deep, and better-performing network (i.e., a teacher model). In the metric learning context, some works explored this idea with a slightly changed formulation: classical knowledge distillation methods use teacher networks to guide the students on individual examples, but metric learning methods exploit similarity relationships between pairs (or groups) of samples. For this, researchers proposed methods that match a number of properties between the embeddings obtained from the teacher and the student models, including the ranks of retrieved samples (Chen et al., 2018b), distances between samples (Park et al., 2019), class likelihood distributions (Han et al., 2019), and absolute positions of embeddings in the latent space (Yu et al., 2019a). With few exceptions (Meseguer-Brocal et al., 2018; Wu & Lerch, 2017), distillation techniques have been largely underexplored in MIR, and we believe that no attempts have been done within VI.

In this set of experiments, we consider MOVE as a teacher model, and our goal is to train from scratch a student model of the same depth but with a lower embedding dimensionality. Our approach is formulated in a deep metric learning setting where the guidance of the teacher model is shaped by the distances between samples (see Figure 6.1(b)). In the experiments we describe next, the weights of the teacher model are frozen, and the weights of the student model are initialized randomly.

**Distance matching —** Perhaps the most intuitive way of guiding the student model is to match the distances obtained from the student with the ones from the teacher, allowing the two models to have different embedding sizes. In our implementation, we pass the samples in each mini-batch to both models, compute in-batch pairwise distances, and use the mean absolute error between the pairwise distance matrices from the teacher model and the student model to train the latter:

$$\mathcal{L}_i^{\mathrm{DM}} = \sum_{j \in \mathsf{J}} \left| E(\mathbf{v}_i^{(\mathrm{s})}, \mathbf{v}_j^{(\mathrm{s})}) - E(\mathbf{v}_i^{(\mathrm{t})}, \mathbf{v}_j^{(\mathrm{t})}) \right|, \tag{6.1}$$

where $E$ is the normalized squared Euclidean distance function defined in Equation 4.4, $\mathsf{J}$ is the set of samples in a mini-batch, and $\mathbf{v}_i^{(\mathrm{s})}$ and $\mathbf{v}_i^{(\mathrm{t})}$ are the embeddings of track $i$ obtained with the student and teacher models, respectively.

**Cluster matching —** Our second knowledge distillation scheme aims to obtain a student model that constructs clusters with both low intra-class and high inter-class distances. Assuming the teacher model holds this desired property, we take advantage of this information to guide the student model. To the best of our knowledge, this distillation criterion has not been explored in previous deep metric learning research.

Our criterion uses internal cluster evaluation metrics (Liu et al., 2010). In the experiments reported here, we use the Davies-Bouldin (DB) index (Davies & Bouldin, 1979), but other cluster evaluation metrics can be used:

$$\mathcal{L}_i^{\mathrm{DB}} = \max_{j \neq i} \left( \frac{\zeta_i + \zeta_j}{E(\mathbf{c}_i, \mathbf{c}_j)} \right), \tag{6.2}$$

where $\zeta_i$ denotes the average intra-class distance, computed with a suitable distance measure $E$, and $\mathbf{c}_i$ denotes the centroid for class $i$. The DB index yields low values for configurations that have low intra-class and high inter-class distances.

In our implementation, we precompute the class centroids using the MOVE embeddings from the entire training set. To match the dimensions of the centroids with the student model embeddings, we train a linear projection simultaneously with the student model. The intra-class and inter-centroid distances are computed using only the samples present in the mini-batch and their respective centroids. DB scores for each class in the mini-batch are computed and then averaged to obtain the final loss value.

#### 6.2.2.4 Latent space reconfiguration

Transfer learning applications take advantage of the strong priors learned by the feature extractor parts of successful, high-capacity models that are trained on large datasets. Inspired by this idea, we hypothesize that by using the feature extractor of a pretrained model, we can obtain a better-structured and lower-dimensional latent space that cannot be obtained by training a randomly initialized model from scratch.

To test this idea, we use the pretrained convolutional layers of MOVE as the feature extractor, remove the final linear layer, and let the network learn a new latent space with a randomly initialized linear layer using a metric learning loss function (see Figure 6.1(c)). Note that the original MOVE model is trained with a triplet loss, meaning that it learned a distance metric parametrized by a neural network. Our idea is to use the nonlinear part of that metric, and "reconfigure" the latent space and the distance metric by optimizing a second loss function (hence the name latent space reconfiguration, or LSR). Our motivation is based on two assumptions: (1) training losses play an important role in shaping the latent space where the embeddings lie, and (2) embeddings with lower dimensionalities may be sufficient to successfully represent semantically meaningful information, as long as the dimensions are effectively utilized.

Although LSR follows the same procedure and shares the underlying idea of transfer learning, we would like to point their differences. Firstly, transfer learning applications, by definition, require different tasks (i.e., source and target tasks) while LSR is applied for the same task. Secondly, the input data distributions for the source and target tasks for transfer learning are generally different, as opposed to the case of LSR where the same data is used for training the base and the reduced models. Lastly, the main purpose of transfer learning can be summarized as improving generalization in a new setting. Focusing on metric learning schemes, the term LSR denotes the process of starting with an already learned distance metric and modifying it to represent the semantic relations in a more compact embedding space.

We consider four loss functions for training the reduced model, which we describe below. The weights of the feature extractor are frozen during the first epoch and updated with a lower learning rate during the rest of the training. Batch normalization is applied after the linear layer as in MOVE (see Section 4.2.3). Apart from using the loss functions described below for LSR, we also use them individually and train models from scratch with the same settings to set baseline models.

**Triplet loss —** We follow the triplet loss formulation denoted in Equation 4.3. Distances among vectors are computed using the normalized squared Euclidean distance $E$ as specified in Equation 4.4:

$$\mathcal{L}^{\text{Triplet}}_{\text{A, P, N}} = \max\left(E\left(\mathbf{v}_{\text{A}}, \mathbf{v}_{\text{P}}\right) - E\left(\mathbf{v}_{\text{A}}, \mathbf{v}_{\text{N}}\right) + m, 0\right), \tag{6.3}$$

where $\mathbf{v}_{\text{A}}$ corresponds to the anchor, $\mathbf{v}_{\text{P}}$ to the positive sample, $\mathbf{v}_{\text{N}}$ to the negative sample, and $m = 1$ is a margin hyperparameter. For selecting which triplets to use, we follow the hard-positive, hard-negative mining strategy explained in Section 4.2.3.

**ProxyNCA loss —** ProxyNCA loss was introduced by Movshovitz-Attias et al. (2017) as a nonlinear extension of the neighborhood component analysis (NCA; Goldberger et al., 2004). Except being parametrized by nonlinear deep learning models, another improvement ProxyNCA introduced over NCA is the concept of proxy vectors per class. With this, distances are computed between items and proxy vectors rather than between pairs of items as in the original NCA formulation. The authors reported that this significantly reduces the amount of computation and also facilitates a faster convergence.

Our implementation of ProxyNCA loss (Movshovitz-Attias et al., 2017) also uses the normalized squared Euclidean distance $E$ from Equation 4.4. Every class in our training set is represented with one proxy vector that is initialized randomly and trained simultaneously with the model parameters. In mathematical notation, the ProxyNCA loss can be expressed as:

$$\mathcal{L}^{\text{P}}_i = -\log\left(\frac{\exp(-E(\mathbf{v}_i, \mathbf{y}))}{\sum_{\mathbf{z}\in \mathsf{Z}_i} \exp(-E(\mathbf{v}_i, \mathbf{z}))}\right), \tag{6.4}$$

where $\mathbf{y} \in \mathbb{R}^d$ denotes the proxy vector for the class of $\mathbf{v}_i$ and $\mathsf{Z}_i$ denotes the set of proxies for all the classes different than the one of item $i$.

**NormalizedSoftmax loss —** Zhai & Wu (2019) introduced the NormalizedSoftmax loss and showed that the regular classification setting provides a competitive baseline against training a model with a metric learning setting. In multiclass classification settings, normally, logits are computed with a dot product between the representation from the penultimate layer of the network and the weights of the final linear layer. The reason why we call this loss function "normalized" is based on the fact that the logits are computed using the cosine similarity, which can be seen as a dot product between L2-normalized vectors.

As proposed by Zhai & Wu (2019), we implement this function using the cosine distance. We randomly initialize one proxy per class and update their parameters at each training step. We use

$$\mathcal{L}^{\text{N}}_i = -\log\left(\frac{\exp(\langle \mathbf{v}_i, \mathbf{y}\rangle/\tau)}{\sum_{\mathbf{z}\in \mathsf{Z}} \exp(\langle \mathbf{v}_i, \mathbf{z}\rangle/\tau)}\right), \tag{6.5}$$

where $\langle\ \rangle$ denotes cosine similarity, $\mathbf{y} \in \mathbb{R}^d$ the proxy for the positive class, $\mathsf{Z}$ the set of proxies for all classes, and $\tau = 0.05$ the temperature parameter.

**Group loss —** Group loss was introduced by Elezi et al. (2020) in an attempt to facilitate classification-based training by incorporating pairwise similarities between items into the process. This is done by updating the initial likelihoods, computed with a softmax function applied on logits, using replicator dynamics that help similar items to have a high likelihood of being in the same class.

Following the approach of Elezi et al. (2020), we use Pearson's correlation coefficient as the similarity metric and replace the negative correlation scores with 0. We perform three iterations for refining the class probabilities and, unlike the original implementation, we select one anchor per class in each mini-batch. The main loss is regular cross-entropy:

$$\mathcal{L}_i^{\mathrm{G}} = -\log\left(\frac{\exp(q_i^{(c^*)})}{\sum_{c\in\mathsf{C}}\exp(q_i^{(c)})}\right), \tag{6.6}$$

where $q_i^{(c)}$ denotes the logit of sample $i$ with respect to the class $c$, $c^*$ denotes the correct class for sample $i$, and $\mathsf{C}$ denotes the set of all classes in the training set.

### 6.2.3 Training details

We use the Da-TACOS training set (see Section 3.2) for developing the models. To find a suitable learning rate and an optimizer for each experiment, we perform a grid search over both SGD and Ranger (Wright, 2019) optimizers and initial learning rates in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, using our validation set. The full training lasts for 70 epochs, and we decrease the learning rate by a factor of 10 at epochs 50 and 60. We save the model weights that result in the best performance on the validation set. The remaining training details and design decisions follow the ones used for training MOVE (see Section 4.2.3). The hyperparameter values used for each experiment can be found in the project repository (see Appendix C).

## 6.3 Results

### 6.3.1 Evaluation methods

All the models are evaluated on the Da-TACOS benchmark subset (see Section 3.2), which contains a nonintersecting set of cliques with respect to our training and validation data. As done previously, we report the experimental results using MAP and MR1 metrics (see Section 2.5.2).

### 6.3.2 Embedding distillation experiments

Table 6.1 presents the exhaustive list of results for the methods described in Section 6.2. The baseline results (top block) show that when training from scratch, chan-

| Method | *d* | | | |
|---|---|---|---|---|
| | 128 | 256 | 512 | 2048 |
| *Baselines (no reduction, training from scratch)* | | | | |
| Triplet | 0.459 | 0.469 | 0.478 | 0.487 |
| ProxyNCA | 0.168 | 0.185 | 0.212 | 0.206 |
| NormalizedSoftmax | 0.445 | 0.470 | 0.475 | 0.422 |
| Group | 0.265 | 0.271 | 0.269 | 0.271 |
| *Unsupervised* | | | | |
| PCA | 0.494 | **0.507** | **0.507** | **0.507** |
| ICA | 0.456 | 0.425 | n/a | n/a |
| GRP | 0.429 | 0.465 | 0.485 | **0.502** |
| *Knowledge distillation* | | | | |
| Distance matching + Triplet | 0.492 | **0.499** | **0.503** | **0.500** |
| Cluster matching + Triplet | 0.424 | 0.471 | 0.465 | 0.455 |
| *Latent space reconfiguration* | | | | |
| Triplet | 0.485 | 0.491 | 0.494 | **0.506** |
| ProxyNCA | 0.424 | 0.465 | 0.485 | **0.502** |
| NormalizedSoftmax | **0.513** | **0.524** | **0.525** | **0.525** |
| Group | 0.465 | 0.483 | **0.495** | **0.511** |

**Table 6.1:** MAP for different embedding sizes $d$ when training from scratch (top) and when using pretrained models and embedding distillation (middle-bottom). MAPs for the original MOVE-4k and MOVE-16k baselines are 0.495 and 0.507, respectively (values equal to or above MOVE-4k are highlighted in bold).

ging the loss function of a network causes significant accuracy differences. It should be noted that all alternative losses we consider achieve state-of-the-art performances in computer vision datasets. Nevertheless, our results suggest that they may not generalize across other types of data or tasks, or that they may be oversensitive to hyperparameters or architectural decisions.

For unsupervised dimensionality reduction (second block of Table 6.1), we see that PCA successfully projected the information contained in MOVE embeddings, even when using 256 dimensions. This suggests that although achieving a highly competitive performance, MOVE embeddings contain redundant information that can be drastically compressed. GRP reached a similar performance as PCA with $d = 2048$, but the resulting performance decreased when using lower-dimensional embeddings.

The initial experiments on pruning reached the same performance as MOVE after one iteration, that is, after reducing the dimensionality by 50%. However, further pruning iterations drastically decreased MAP, up to the point of yielding useless embeddings. Therefore, we decided to stop iterating and not report the corresponding results.

Among the considered knowledge distillation techniques (third block, Table 6.1), the

| | *d* | MAP | MR1 |
|---|---|---|---|
| *Individual systems* | | | |
| 2DFTM (Bertin-Mahieux & Ellis, 2012) | 50 | 0.275 | 155 |
| SiMPle (Silva et al., 2016) | 2.2 k | 0.332 | 142 |
| Dmax (Chen et al., 2018a) | 5.5 k | 0.398 | 167 |
| Qmax (Serrà et al., 2009a) | 5.5 k | 0.437 | 138 |
| Qmax* (Serrà et al., 2009b) | 5.5 k | 0.445 | 130 |
| MOVE-16 k | 16 k | 0.507 | 46 |
| **Re-MOVE** | **256** | **0.524** | **43** |
| *Ensemble and fusion systems* | | | |
| SNF (Qmax & Dmax) (Chen et al., 2018a) | 5.5 k | 0.533 | 139 |
| Avg. (MOVE-512 & MICE w/ dMel) | 1 k | 0.621 | 32 |
| LF-c (MOVE-512 & MICE w/ dMel) | 512 | 0.635 | 30 |
| SNF (Qmax & MOVE-16 k) | 21.5 k | 0.651 | 62 |
| SNF (MOVE-16 k & **Re-MOVE**) | 16.3 k | 0.595 | 65 |
| SNF (Qmax & **Re-MOVE**) | 5.7 k | 0.660 | 61 |

**Table 6.2:** Comparison of Re-MOVE and other state-of-the-art VI systems on the Da-TACOS benchmark subset. When not explicit, embedding sizes *d* are estimated for a track duration of 3.5 min (see Section 6.3.3). MOVE-512 and MOVE-16 k denote MOVE models that produce embeddings of size 512 and 16 k, respectively. Results for the proposed system are highlighted in bold.

additional distance matching loss clearly increases the model performance compared with the case where the model is trained from scratch. However, the same advantage is not observed with cluster matching using DB loss. We hypothesize that this may be related to training an extra linear projection for compressing the centroid embeddings to match the size of the embeddings obtained with the student model.

Latent space reconfiguration results seem to justify our hypothesis regarding the use of strong priors of a pretrained feature extractor (last block, Table 6.1). All the considered alternatives outperform their baseline counterparts. Moreover, we find that using probabilistic losses such as NormalizedSoftmax and Group loss for latent space reconfiguration even outperform the original model while reducing the embedding size by a large margin ($128/16000 = 0.8\%$). Note that in addition to these advantages, latent space reconfiguration does not suffer from the setbacks of network pruning and knowledge distillation methods, namely training a model for multiple iterations and using two models simultaneously during training, respectively.

### 6.3.3 Comparison with the state of the art

Lastly, Table 6.2 compares the best result obtained in this chapter with state-of-the-art methods. The second column, *d*, shows the size of the smallest representation (per

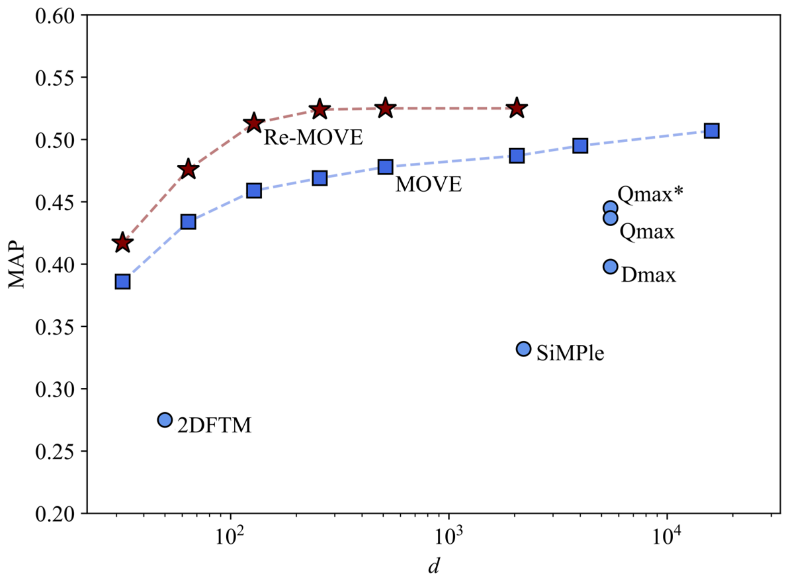


**Figure 6.2:** MAP with respect to embedding dimensionality $d$ for Re-MOVE (red stars), MOVE (blue squares), and other existing approaches (blue circles). Notice the logarithmic axis.

track) required for each method to estimate pairwise similarities (equivalent to the embedding dimensionality). As the results for the systems excluding MOVE and Re-MOVE are computed with the publicly available acoss library (see Section 3.4), we use those implementations for estimating the embedding sizes[38]. As the sizes of some representations depend on the track duration or tempo, we use 3.5 min and 102 bpm estimates, which correspond to the average track duration and bpm of the tracks in the Da-TACOS benchmark set, respectively.

Re-MOVE, which stands for "reduced MOVE," denotes the model trained with latent space reconfiguration using NormalizedSoftmax. With $d = 256$, it demonstrates relative performance increases of 3% and 18% when compared with MOVE and Qmax* systems, respectively (Table 6.2). Also, the ensemble of Qmax and Re-MOVE using similarity network fusion yields a better performance compared to its alternatives, including the ensemble of Qmax and MOVE. We also find that Re-MOVE improved over MOVE for a wide range of dimensionalities $d \in [32, 2048]$ (Figure 6.2). Along with its state-of-the-art performance, Re-MOVE provides a crucial advantage in terms of scalability, which positions it as the most viable system from a practical point of view compared to the considered alternatives.

[38]For 2DFTM, the acoss library uses a 450-dimensional embedding while the authors apply PCA to reduce the dimensionality to 50.

### 6.3.4 Error analysis

We conclude this section with an error analysis using Re-MOVE, as we have done for MOVE (see Section 4.3.5). We randomly select non-version pairs that result in low distances and version pairs that result in high distances. Our analysis confirms the issues we have mentioned in Section 4.3.5, but we do not expand that list of issues with this analysis. There are cases where Re-MOVE performs better than MOVE, but we cannot see any pattern behind those cases. In total, there are 7091 queries with higher AP scores for Re-MOVE compared to MOVE and 5338 queries for the opposite. Considering that we use the same training data and a similar training strategy for Re-MOVE as we have done for MOVE, it is not surprising that the failed cases follow similar patterns, and the differences between them may not result from solving a particular problem of MOVE.

## 6.4 Conclusion

In this chapter, we have introduced a set of techniques for reducing the embedding sizes of existing VI systems, which we have considered under the name embedding distillation. We have claimed that by using a pretrained and high-capacity network, it is possible to train a second network that yields smaller embedding vectors without a decrease in performance. To investigate this idea, we have studied a wide range of techniques, including classical dimensionality reduction, neural network pruning, and knowledge distillation methods. Moreover, we have introduced latent space reconfiguration, which is a technique that builds upon the nonlinear part of a distance metric learned by a pretrained network to construct a compact latent space with fewer dimensions. Our results have shown the possibility of reducing the embedding dimensionality of a model while maintaining, or even surpassing, its performance. With this, we have obtained the Re-MOVE model, which improves upon the scalability aspect of our baseline model, MOVE.

With the methods that improve accuracy (see Chapter 5) and scalability (described in this chapter), we have aimed to take a step toward bridging the gap between academic research and industrial applications. Although identification performance and processing times are important requirements for industrial use cases, another point that needs to be carefully investigated is the algorithmic bias in VI systems. In the next chapter, we acknowledge the impact VI systems may have on musicians and analyze whether current systems favor certain musicians over others, to foresee and mitigate any undesired consequences.

Chapter 7

# Exploring the Algorithmic Bias in Version Identification

## 7.1 Introduction

Developing automatic systems for industrial use cases has been one of the main goals of VI research since its inception, and it is one of the main goals of this dissertation in particular. To this end, a great number of methods have been dedicated to improving the systems from accuracy and scalability perspectives. Thanks to the recent developments in VI, to which this dissertation has also contributed, nowadays, VI systems start being used in commercial applications, ranging from music recognition to music discovery. Importantly, such usage can have an impact on various stakeholders in the music ecosystem, especially regarding recognition and financial benefits.

Digital rights management (DRM) for musical recordings is a complex ecosystem, where multiple types of royalties circulate between a large amount of stakeholders[39]. When artists perform a version of an existing composition, they are obliged to pay "mechanical royalties" to the holders of "the composition copyright." With the help of VI systems, detecting such cases can be automated. However, VI systems cannot match a recording to a composition directly; instead, they match an unknown recording to the known recordings of a composition. Therefore, we can identify three main parties involved in the process: the performing artists of the unknown (or query) recording, the performing artists of the known (or reference) recording, and the composers of the composition. Taking into account the impact that VI systems may have over such artists and composers, they can be considered as socio-technical systems (Baxter & Sommerville, 2011; Selbst et al., 2019) rather than isolated technologies for such use cases.

Fairness and transparency is an emerging field that studies the societal implications of algorithmic systems (Olteanu et al., 2021; Mehrabi et al., 2021). Any existing

[39] https://www.taxi.com/transmitter/1906/how-does-the-music-industry-work/

biases in such systems may lead to favor certain individuals or groups over others, which may result in "unfair" outcomes (Fletcher et al., 2021). Although algorithmic decisions and their impact are closely related concepts, measuring fairness is a domain- and context-dependent process, mainly due to the fact that investigating societal impact requires a certain legal, cultural, and ethical framework. However, quantifying algorithmic biases is a useful step for mitigating potential unfair decisions. In music technology research, recent works have addressed potential issues in music recommendation from both individual and group fairness perspectives (Chouldechova & Roth, 2020; Räz, 2021) by studying gender imbalance (Ferraro et al., 2021; Shakespeare et al., 2020) and playlist diversity (Porcaro & Gómez, 2019; Robinson et al., 2020). In a recommendation context, a fairness study may investigate whether two potentially impacted groups are subject to the same exposure or not, assuming more exposure leads to larger financial gains. Nonetheless, in a music recognition task like VI, measuring fairness with respect to such exposure may not be meaningful, as the main goal of VI systems is to correctly detect items that are objectively linked to each other as "versions of the same composition." However, this does not imply that tasks like VI, which have objectively assigned labels, do not require any studies about fairness or algorithmic biases. Potentially, a discrepancy in the identification performance of such VI systems with respect to some characteristics (e.g., demographics) of the involved parties (e.g., performing artists or composers) may put some musicians in a favored position (e.g., financially) compared to others. Therefore, an examination into whether VI systems inherit any form of bias is necessary to study the potential implications of such systems in the music ecosystem.

In this chapter, we propose a framework for investigating the performance discrepancies of VI systems across a selected set of attributes, in order to explore their impact on relevant parties. Although our main motivation is to raise awareness for developing fair VI systems, we cannot reach any conclusions on the fairness qualities of existing systems since such fairness measurements are context-dependent and we do not have access to a system that is interacting with the parties we consider. Instead, we aim to quantify the algorithmic bias by investigating the discrepancies in identification performances of five state-of-the-art VI systems of different characteristics on pairs of potentially impacted groups, mimicking a group fairness paradigm (Räz, 2021). We categorize such groups using the three aforementioned parties (performing artists and composers) and six relevant side attributes (gender, popularity, country, language, year, and prevalence). To carry out our analyses, we developed the VI-Bias dataset, for which we collected metadata in terms of the attributes we investigate and assigned them to tracks of an existing dataset. We then gather performance data on pairs of groups and analyze the existence of potential discrepancies using the Kolmogorov-Smirnov test. Our results from a total of 115 experiments suggest that on the one hand, the characteristics of a VI system may play a role in favoring a certain group over its counterpart but that on the other hand, there also exist cases where all the VI systems favor the same or no group. After presenting the results, we also share our hypotheses on the possible reasons for a subset of the observed disparities. To facilitate future research, we share the instructions on how to obtain the VI-Bias dataset and the evaluation code at the project repository[40] (see Appendix C).

This chapter is based on Yesiler et al. (2022).

## 7.2 Methods

### 7.2.1 Systems

In this study, we consider a range of VI systems with different characteristics, in terms of whether they are learning- or rule-based, or whether they use melody- or chroma-based inputs. With this, we aim to better understand and discuss the causes of possible performance differences.

**Qmax —** Qmax (Serrà et al., 2009a) is a rule-based system that estimates similarities between pairs of tracks using an elaborate local alignment scheme, which results in a good performance but suffers from high computational cost. Nevertheless, it was the best-performing VI system for more than a decade, until recently. For our experiments, we use the implementation included in Essentia (Bogdanov et al., 2013), with cremaPCP features (see Section 3.5).

**MOVE —** MOVE is a deep learning–based system that features musically motivated design decisions to bring inductive biases to the model (see Chapter 4). It also uses cremaPCP features as input and transforms them into fixed-size embedding vectors, regardless of the input duration.

**MICE-M and MICE-C —** MICE is a deep learning–based system that combines the design decisions of the model introduced by Doras & Peeters (2019) and MOVE (see Chapter 5). Due to its input-agnostic design, MICE can be trained and used with any input feature suitable for VI. Hence, we consider two variants of the algorithm: MICE-M, which uses dominant melody features as input, and MICE-C, which uses cremaPCP features.

**LF-c —** LF-c is a deep learning–based system designed to investigate the performance gains of combining models that use complementary features (see Chapter 5). It concatenates the embeddings obtained by MOVE and MICE-M and projects these into a new space, resulting in new embeddings.

### 7.2.2 Attributes

For each of the attributes presented below, we categorize recordings into two groups, inspired by group fairness studies where the majority of literature considers binary groups (Räz, 2021). Considering various parties involved in VI, these groups can be

[40]https://github.com/furkanyesiler/vi_bias

created with respect to (1) the artist or the performance of the query track, (2) the artist or the performance of the reference or “original” track, and (3) the composer or the composition.

In the case of attributes that relate to persons (e.g., gender) rather than recordings (e.g., language), if the artist of a recording is a person, we assign the label for that recording as that of that person. If the artist is a band, we then collect the labels for all the current and past band members. To have clear distinctions between groups, we do not consider bands that have mixed labels (i.e., where some members belong to one group while the others belong to the other group). We also exclude from our analyses the cases where we cannot find a label for even a single member of a band.

We study differences in system performances by using binary labels (group 1, or G1, and group 2, or G2) for each attribute. If not stated otherwise, such labels are derived with respect to the query recordings or their artists (-Q), the reference recordings or their artists (-R), and the compositions or their composers (-C). We also analyze the performance differences where the query and the reference recordings or their artists belong to the same vs. different groups (-SD). In the case where they belong to different groups, we also check if the direction of the change (i.e., the reference in G1 and the query in G2 or the reference in G2 and the query in G1) has any effect on the performance (-D12). A summary of all the experiments and the considered attributes can be seen in Table 7.1.

Note that for the -SD experiments, G1 denotes the cases where queries and references belong to the same group (e.g., queries and references from male artists) while G2 denotes that they belong to different groups (e.g., queries from male artists and references from female artists). For the -D12 experiments, G1 denotes the cases where queries and references are in the first and the second groups of the -Q experiments, respectively (e.g., queries from male artists and references from female artists); and G2 denotes cases where the queries and references are in the second and the first groups of the -Q experiments, respectively (e.g., queries from female artists and references from male artists).

**Gender —** The two groups we consider for gender are male (G1) and female (G2), and the data is gathered from MusicBrainz (MB; Swartz, 2002). The experiments we perform for this attribute are denoted as G-Q (queries from male vs. female artists), G-R (references from male vs. female artists), G-C (compositions from male vs. female composers), G-SD (queries and references from the same vs. different gender groups), and G-D12 (queries from male and references from female artists vs. queries from female and references from male artists).

**Popularity —** For this attribute, we categorize the recordings as popular (G1) and “not so popular” (G2). We consider an artist (whether a person or a band) as popular if they ever had a number-one selling single in the top-10 music markets found in the

| Feature | Query (-Q) | Ref. (-R) | Comp. (-C) | Same-Dif. (-SD) | Dif. G (-D12) |
|---|---|---|---|---|---|
| Gender (G-) | 1**2345** | 1**2345** | 12**3**4**5** | 1**2345** | 12345 |
| Popularity (P-) | 1**2345** | **12345** | 12345 | **12345** | 12345 |
| Country (C-) | 12345 | 12345 | 12**3**4**5** | 12345 | 12345 |
| Language (L-) | 12**3**4**5** | 12**3**4**5** | - | - | - |
| Year (Y-) | **12345** | **12345** | - | **12345** | - |
| Prevalence (V-) | **1**2**3**45 | 1**23**45 | **12**3**45** | - | - |

**Table 7.1:** Summary of the experiments. Numbers indicate the considered systems: Qmax (1), MOVE (2), MICE-M (3), MICE-C (4), and LF-c (5). The bold and underlined text indicates that the results obtained with the corresponding system show a significant difference between G1 and G2.

IFPI's Global Music Report 2021[41] (excluding South Korea and China). By including European and Far-eastern music markets, we aim to mitigate bias toward artists from the United States. In the case of composers, we consider them popular if a popular artist (as defined above) ever played a version of any of their compositions. The number-one selling singles data is collected from Wikipedia, and the data for all the compositions of every composer is collected from MB. Although there are many ways to define popularity, our decision is based on the fact that number-one selling single data is the only data that can be found consistently (as opposed to, e.g., top-10 singles) for all the aforementioned music markets on Wikipedia, which in itself suggests a distinction between the number-one selling artists and the rest. The experiments we perform for this attribute are denoted as P-Q, P-R, P-C, P-SD, and P-D12.

**Country —** For studying the effect of the country, we divide recordings by the ones whose artists or composers are from the United States or the United Kingdom (G1), and the rest (G2), as the tracks in our dataset consist of mostly genres originated in music traditions centered around those countries. In MB, there exist two annotations for the country: "area," the place where the artist currently resides, and the "begin-area," the place where the artist is originally from. In cases where an artist has both annotations, we use the begin-area for our analysis. The experiments we perform for this attribute are denoted as C-Q, C-R, C-C, C-SD, and C-D12.

**Language —** The two categories we consider for the language are English (G1) and other (G2). The language data is gathered from SecondHandSongs (SHS). The experiments for this attribute are denoted as L-Q and L-R. We exclude the other experiments since we cannot obtain the data for the language of compositions (L-C) and we do not have enough samples for analyzing L-SD and L-D12.

**Year —** For this attribute, we divide the recordings into two groups by whether they are from the year 2000 or after (G1) or from before (G2). Although this threshold can

[41] https://gmr2021.ifpi.org/report

| Exp. | G1 | G2 | Exp. | G1 | G2 |
|---|---|---|---|---|---|
| G-Q | 8,067 | 3,296 | C-C | 13,200 | 3,535 |
| G-R | 11,976 | 2,954 | C-SD | 424 | 109 |
| G-C | 16,840 | 1,093 | C-D12 | 26 | 83 |
| G-SD | 5,943 | 2,040 | L-Q | 18,765 | 2,069 |
| G-D12 | 603 | 1,437 | L-R | 19,487 | 2,172 |
| P-Q | 2,964 | 17,200 | Y-Q | 9,310 | 12,019 |
| P-R | 6,432 | 13,505 | Y-R | 2,579 | 18,517 |
| P-C | 3,171 | 1,255 | Y-SD | 7,737 | 11,129 |
| P-SD | 11,638 | 6,360 | V-Q | 11,025 | 11,342 |
| P-D12 | 1,608 | 4,752 | V-R | 10,211 | 9,882 |
| C-Q | 1,879 | 638 | V-C | 9,024 | 13,388 |
| C-R | 2,967 | 594 | | | |

**Table 7.2:** Sample sizes per group per experiment.

be set in many different ways, we use the year 2000 since the peer-to-peer networks that changed the music ecosystem drastically became popular by then. The release year data is gathered from SHS. For this attribute, the experiments we perform are denoted as `Y-Q` and `Y-R`. We also analyze the system performances in cases where the year gap between the query and the reference recordings is less than 10 years (G1) vs. greater than or equal to 10 years (G2) and denote the experiment as `Y-SD`.

**Prevalence —** This last attribute simply characterizes either how prevalent it is for a composition to have a version, quantified by the number of versions a composition has, or how prevalent it is for an artist to perform versions of other artists' tracks, quantified by the number of versions an artist performed. We include this attribute in our analyses to see whether tracks that have been played by many artists (or artists that have played tracks from many others) are in an advantageous position in terms of DRM or not. The data we use to form the categories is collected from SHS. For the artist prevalence experiments (`V-Q` and `V-R`), the two groups we consider are the artists who performed less than or equal to 25 versions (G1) and more than 25 versions (G2). For the composition prevalence experiment (`V-C`), the two groups we consider are the compositions that have less than or equal to 5 versions (G1) and more than 5 versions (G2). Thresholds 25 and 5 are chosen as they are the median values of their respective distributions.

### 7.2.3 Dataset and evaluation

#### 7.2.3.1 Dataset

We perform our analyses using a subset of the SHS4- dataset (Doras & Peeters, 2019), for which we populate the metadata and label annotations. We publicly release this subset under the name VI-Bias, with Creative Commons BY-NC-SA 4.0 license[42]. The reference set of VI-Bias includes only one recording per composition, and we specifically chose them as the original recordings for all the compositions (as stated in SHS). Therefore, every query has exactly one correct item in the reference set. The reason for this decision is simply to imitate industrial cases where it is common to have only the original recording in the reference set. However, there can be multiple queries that have the same reference recording as the correct item. The total number of queries and references in VI-Bias is 22,428 and 15,293, respectively. All the deep learning–based models are trained using the SHS5+ dataset (Doras & Peeters, 2019). For the training details of the considered systems, see Chapters 4 and 5.

#### 7.2.3.2 Evaluation

We assess system performances using the reciprocal rank metric, which we denote by $\psi$. We motivate this choice by the fact that $\psi$ penalizes the differences in lower ranks more than the differences in higher ranks, which we consider as a good proxy for real use cases (e.g., the difference between the correct answer being at rank 1 or 11 should have more impact than the difference of it being at rank 31 or 41). Note that since there is only one correct item in the reference collection for each query, $\psi$ corresponds to the same score as average precision.

For each group in our study, we collect $\psi$ scores for all the items belonging to that group and use them to form performance distributions. To compare the obtained distributions of two groups, we use the two-sample Kolmogorov-Smirnov (KS) test (Massey, 1951), which has several characteristics that are desirable for our setup (e.g., it is a nonparametric exact test, and it is not sensitive to imbalanced sample sizes between groups). The null hypothesis of the KS test is that the samples are drawn from the same distribution (i.e., that the underlying distribution is the same). To reject the null hypothesis, we consider the threshold of 0.05 for the $p$-value; however, since we test multiple hypotheses, we apply the Holm-Bonferroni correction (Holm, 1979). For computing the KS test and plotting rank distributions (see Figure 7.1), we collapse all the ranks above 10, to remove potential differences in the tails (those are the high ranks that are typically not going to be considered by practitioners). For each analysis, we report sample sizes (see Table 7.2) and the mean $\psi$ scores, $\bar{\psi}$, for each group.

## 7.3 Results

### 7.3.1 Main findings

Table 7.3 presents the detailed list of results for all the experiments and the considered VI systems, and Figure 7.1 presents rank distribution illustrations for a selected set

[42]https://creativecommons.org/licenses/by-nc-sa/4.0/

| | Qmax ($\bar{\psi}$=0.42) | | | MOVE ($\bar{\psi}$=0.63) | | | MICE-M ($\bar{\psi}$=0.50) | | | MICE-C ($\bar{\psi}$=0.59) | | | LF-c ($\bar{\psi}$=0.75) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Exp. | $p$ | $\bar{\psi}$-G1 | $\bar{\psi}$-G2 | $p$ | $\bar{\psi}$-G1 | $\bar{\psi}$-G2 | $p$ | $\bar{\psi}$-G1 | $\bar{\psi}$-G2 | $p$ | $\bar{\psi}$-G1 | $\bar{\psi}$-G2 | $p$ | $\bar{\psi}$-G1 | $\bar{\psi}$-G2 |
| G-Q | <0.01 | 0.41 | 0.45 | **<0.01*** | 0.63 | 0.70 | **<0.01*** | 0.46 | 0.61 | **<0.01*** | 0.58 | 0.65 | **<0.01*** | 0.73 | 0.81 |
| G-R | 0.41 | 0.42 | 0.44 | **<0.01*** | 0.63 | 0.69 | **<0.01*** | 0.48 | 0.64 | **<0.01*** | 0.59 | 0.65 | **<0.01*** | 0.74 | 0.83 |
| G-C | 0.19 | 0.42 | 0.39 | <0.01 | 0.63 | 0.67 | **<0.01*** | 0.49 | 0.60 | <0.01 | 0.58 | 0.63 | **<0.01*** | 0.74 | 0.80 |
| G-SD | 0.14 | 0.44 | 0.42 | **<0.01*** | 0.65 | 0.70 | **<0.01*** | 0.51 | 0.56 | **<0.01*** | 0.60 | 0.65 | **<0.01*** | 0.74 | 0.81 |
| G-D12 | 0.05 | 0.38 | 0.44 | 1.00 | 0.70 | 0.71 | 0.09 | 0.52 | 0.57 | 1.00 | 0.65 | 0.65 | 1.00 | 0.80 | 0.81 |
| P-Q | 0.06 | 0.44 | 0.42 | **<0.01*** | 0.71 | 0.62 | **<0.01*** | 0.58 | 0.49 | **<0.01*** | 0.66 | 0.57 | **<0.01*** | 0.82 | 0.74 |
| P-R | **<0.01*** | 0.47 | 0.40 | **<0.01*** | 0.70 | 0.60 | **<0.01*** | 0.55 | 0.48 | **<0.01*** | 0.66 | 0.56 | **<0.01*** | 0.80 | 0.72 |
| P-C | 0.57 | 0.41 | 0.39 | <0.01 | 0.63 | 0.58 | 0.83 | 0.50 | 0.48 | 0.01 | 0.59 | 0.54 | 0.38 | 0.75 | 0.73 |
| P-SD | **<0.01*** | 0.41 | 0.45 | **<0.01*** | 0.61 | 0.69 | **<0.01*** | 0.48 | 0.53 | **<0.01*** | 0.57 | 0.64 | **<0.01*** | 0.73 | 0.79 |
| P-D12 | 0.02 | 0.43 | 0.46 | 1.00 | 0.69 | 0.69 | 0.03 | 0.55 | 0.53 | 0.99 | 0.63 | 0.64 | 1.00 | 0.79 | 0.79 |
| C-Q | 0.96 | 0.42 | 0.41 | 1.00 | 0.61 | 0.61 | 0.98 | 0.43 | 0.45 | 0.78 | 0.55 | 0.57 | 0.54 | 0.69 | 0.73 |
| C-R | 0.72 | 0.43 | 0.44 | 0.34 | 0.61 | 0.65 | 0.78 | 0.43 | 0.45 | 0.02 | 0.57 | 0.63 | 0.89 | 0.71 | 0.73 |
| C-C | 0.62 | 0.41 | 0.42 | <0.01 | 0.63 | 0.65 | **<0.01*** | 0.49 | 0.56 | <0.01 | 0.58 | 0.61 | **<0.01*** | 0.74 | 0.79 |
| C-SD | 1.00 | 0.45 | 0.47 | 0.30 | 0.65 | 0.58 | 0.88 | 0.42 | 0.39 | 0.63 | 0.60 | 0.56 | 0.66 | 0.72 | 0.69 |
| C-D12 | 0.66 | 0.59 | 0.44 | 0.22 | 0.73 | 0.54 | 1.00 | 0.40 | 0.39 | 0.54 | 0.67 | 0.53 | 1.00 | 0.69 | 0.69 |
| L-Q | 1.00 | 0.42 | 0.41 | 0.07 | 0.64 | 0.66 | **<0.01*** | 0.51 | 0.64 | 0.12 | 0.59 | 0.61 | **<0.01*** | 0.75 | 0.82 |
| L-R | 0.98 | 0.42 | 0.41 | 0.06 | 0.64 | 0.66 | **<0.01*** | 0.50 | 0.63 | 0.07 | 0.59 | 0.61 | **<0.01*** | 0.75 | 0.82 |
| Y-Q | **<0.01*** | 0.46 | 0.39 | **<0.01*** | 0.66 | 0.61 | **<0.01*** | 0.52 | 0.48 | **<0.01*** | 0.62 | 0.56 | **<0.01*** | 0.77 | 0.73 |
| Y-R | **<0.01*** | 0.48 | 0.41 | **<0.01*** | 0.67 | 0.63 | **<0.01*** | 0.64 | 0.48 | **<0.01*** | 0.65 | 0.58 | **<0.01*** | 0.82 | 0.73 |
| Y-SD | **<0.01*** | 0.44 | 0.40 | **<0.01*** | 0.67 | 0.60 | **<0.01*** | 0.53 | 0.45 | **<0.01*** | 0.63 | 0.55 | **<0.01*** | 0.78 | 0.71 |
| PR-Q | **<0.01*** | 0.43 | 0.40 | 0.16 | 0.62 | 0.64 | **<0.01*** | 0.49 | 0.52 | 0.37 | 0.58 | 0.59 | 0.18 | 0.74 | 0.75 |
| PR-R | 0.11 | 0.43 | 0.41 | **<0.01*** | 0.62 | 0.65 | **<0.01*** | 0.49 | 0.52 | 0.04 | 0.58 | 0.60 | <0.01 | 0.74 | 0.76 |
| PR-C | **<0.01*** | 0.45 | 0.40 | **<0.01*** | 0.65 | 0.61 | 1.00 | 0.50 | 0.50 | **<0.01*** | 0.61 | 0.57 | **<0.01*** | 0.76 | 0.74 |

**Table 7.3:** Detailed results. For each system and experiment, we report the raw $p$-value of the KS test and the mean reciprocal rank of groups 1 and 2 ($\bar{\psi}$-G1 and $\bar{\psi}$-G2). Bold text with * denotes statistical significance after the Holm-Bonferroni correction. Overall system performance with the considered dataset is indicated in parenthesis in the top row.

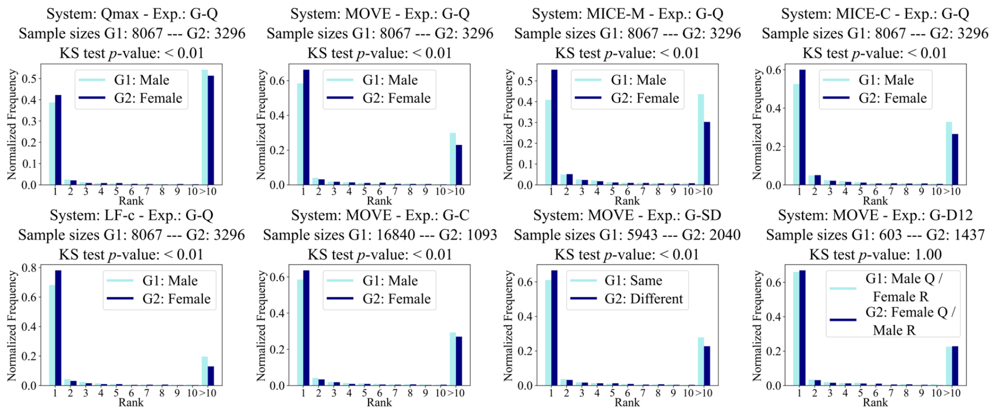


**Figure 7.1:** Rank distributions for a selected set of experiments. The full set of 115 experiments is available in the project repository (see Appendix C).

of experiments. We include the rank distribution illustrations for all the experiments along with the detailed scores in the project repository (see Appendix C). Here, we highlight the main findings with respect to each considered attribute. Possible reasons for the observed disparities are discussed in Section 7.3.2.1.

**Gender —** The results for `G-Q`, `G-R`, and `G-C` suggest that the identification performances of the learning-based models form two distinct distributions for most systems (see Figure 7.1), and performances are significantly higher for the recordings performed or composed by female artists (G2). On the contrary, for the only rule-based system we consider, the obtained *p*-values for `G-Q`, `G-R`, and `G-C` do not suggest a difference between the performance for male (G1) and female (G2) artists/composers. The results for `G-SD` and `G-D12` show that the learning-based systems perform better when the gender of the query and the reference artists are different (G2), and that the direction of the change (female reference to male query, or male reference to female query) does not create a significant difference.

**Popularity —** Results for the popularity attribute suggest that all the systems perform better for queries and references from popular artists (G1); only for `P-Q` using Qmax does not result in a significant difference. However, such a difference between groups is not observed for the composers (`P-C`). Moreover, when the artists of the query and the reference tracks belong to different popularity groups (G2), identification performance is higher (`P-SD`). As with the case with gender, the direction of the change does not create an important difference (`P-D12`).

**Country —** Regarding the results for the country attribute, we do not see any differences between performances with respect to the country of the performing artists (`C-Q` and `C-R`). However, MICE-M and LF-c systems show such a difference with respect to the country of the composer, favoring composers from outside the US/UK (G2) group (`C-C`). We also observe no differences for the cases where the query and the reference recordings belong to the same (G1) vs. different (G2) groups (`C-SD`) and when the direction of the change is different (`C-D12`), but note that the sample sizes for the last two experiments are smaller than the previous ones.

**Language —** The results for the language attribute suggest a performance difference between groups for the systems that use melody-based features (i.e., MICE-M and LF-c) but not for the ones that use only chroma-based features (`L-Q` and `L-R`).

**Year —** All the considered experiments for the year category suggest the same result across all the systems: identification performance for queries and references after 2000 (G1) is higher compared to queries and references before that year (`Y-Q` and `Y-R`). Also, when the year gap between the query and the reference tracks is less than a decade (G1), the systems perform better.

**Prevalence —** We see no pattern behind the differences in the results for query recordings from artists that perform versions of other artists' compositions less (G1) or more (G2) prevalently (`V-Q`): while the performances of the considered groups seem similar for MOVE, MICE-C, and LF-c systems, MICE-M and Qmax favor different groups. For the experiments that compare the same quality for reference recordings (`V-R`), however, we see that the learning-based systems favor the cases where the reference recordings are from artists that perform versions more prevalently (G2) than the others (G1), while Qmax favors the opposite (G1 over G2). Lastly, in terms of comparing compositions with fewer (G1) or more (G2) versions (`V-C`), all the systems except MICE-M suggest a significant difference between the considered groups, favoring compositions with fewer versions (G1) over the others (G2).

### 7.3.2 Discussion

Before entering into any discussion of the results, we shall first remind ourselves of the limitations of our data collection process. Firstly, the data sources we used (MB, SHS, and Wikipedia) contain human-annotated data, but we still would like to point out the possibility of human error. Secondly, since we excluded the cases where we could not find a label for a certain attribute, this may have created a potential bias on the data we use. Lastly, although we implemented many heuristics to obtain the correct data from our sources, we only performed random checks whether we succeeded on that, rather than checking all the obtained annotations one by one. Therefore, there may have been a small amount of metadata matching problems. However, we assume that even if such cases exist, they should not drastically affect our results. In addition to the data collection process, the limitations of creating binary groups might also have an impact on the results. For example, creating such binary groups based on other criteria, or using nonbinary partitions might have changed the outcomes of the presented analyses. Nevertheless, not using binary groups would have forced us to perform a different statistical test, and such tests using multiple categories are less standard and may be less reliable and harder to interpret. Keeping these in mind, we now present our hypotheses on the causes of the observed disparities between groups and discuss the potential fairness implications of the results.

#### 7.3.2.1 Causes of disparities

Overall, we observe that the learning-based systems work better for underrepresented groups (see, for instance, G-Q, G-R, P-Q, and L-Q). Our first hypothesis for this is that the groups with fewer samples may show less variety between versions in terms of the musical characteristics; thus, the identification performance may end up higher. A second reason for this may be the sample sizes of the groups in the training set, but we see that, for example, the gender ratio (G1 vs. G2) in the training set is 2.5, which is not very different than the one of the analysis set, which is 3.2.

When analyzing the attributes separately, we observe that the performances of the learning-based systems are higher for female artists and composers, but also the differences between $\bar{\psi}$ for G1 and G2 are higher for the systems that use melody features. We think that this may be related to the differences between male and female voices and the ability of the melody extraction algorithm to correctly estimate them. Also, looking at the experiments where the query and the reference recordings belong to the same vs. different groups for gender and popularity attributes (G-SD and P-SD), we see that the learning-based systems perform better for the cases where they belong to different groups (G2). We think that when the groups of the query and the reference change, the resulting version may appear more faithful to the original, and this may positively affect system performance.

For the popularity experiments, we see that all the systems tend to perform better for popular artists and composers (G1), though not all the experiments show a significant difference (P-Q and P-C). We hypothesize that this may be due to the input feature extraction algorithms, and the fact that they may have been optimized using tracks from such popular artists. Especially among the learning-based systems, MOVE and MICE-C show larger disparities between groups, and the chord estimation model used for cremaPCP was trained using a dataset including mainly popular tracks.

The results for the language experiments show disparities only for the systems using melody features, with recordings in languages other than English (G2) having better results than recordings in English (G1). A couple of reasons could be that, as mentioned earlier, having fewer samples may cause less variety between versions, and that the phonetic qualities of languages may affect the success of melody estimation algorithms (for example, Italian is argued to be the most suitable language for opera[43]).

In terms of the year attribute, we see that all the systems perform better for queries and references released in the year 2000 or after (G1). Possible reasons for this include the trends in the music creation process, which may affect the degree of variance

[43] https://www.melofonetica.com/italian-is-a-language-built-to-be-sung/

between versions of a composition; the fact that G1 spans over a narrower window of years, which may naturally limit the possibility of having more different versions; and having fewer samples for G1 (as discussed above). For the year gap experiments (`Y-SD`), we argue that when the year gap between versions is longer (G2), it becomes more probable to see versions that are less faithful to the original track, which may cause lower system performances.

#### 7.3.2.2 Considerations for fairness implications

Fairness evaluations are generally applied to systems that already interact with humans, where their direct impact on the considered groups is studied. In our case, however, we do not have access to a deployed VI system nor any real metrics on how the considered groups are affected (e.g., real amount of circulated royalties). Therefore, we now discuss the "potential" implications of the results presented in Section 7.3.1, rather than evaluating a real industrial scenario. Although the following considerations are speculative in nature, they are based on our quantitative analyses, and we believe that not having access to a real industrial system should not preclude research on such problems.

For interpreting the performance differences as fairness outcomes, we take the application of VI in DRM. In our multi-stakeholder scenario, we assume a direct connection between identification performance and the financial impact on various parties: if a query is not correctly identified, the performing artist of the query does not pay and the composer does not receive due royalties. To that extent, the reciprocal rank metric may have opposite meanings for the involved parties. The artist of the query may want a lower identification performance while the composer may desire the opposite.

To better illustrate this point, we can take a look at the gender experiments. We observe that the learning-based systems work better for female artists/composers compared to males. While this implies that female composers are likely to be rewarded more by these systems, in contrast, female artists that perform a version of an existing composition (i.e., artists of the queries) are likely to pay more royalties. Therefore, we here have a case where interpreting the fairness outcomes should be considered independently for all the involved parties, as a result of having a multisided structure.

Lastly, we observe that both the learning- and the rule-based systems show performance disparities for certain groups. Specifically, the learning-based systems show disparities for 54.4% of the cases while this ratio is only 30.4% for the rule-based system. While this clearly presents a disadvantage for the learning-based systems, their advantages regarding accuracy and scalability make them irreplaceable for the industry. Therefore, to make sure such systems do not create any unfair conditions that favor certain groups, evaluating VI systems that may have a real impact on musicians should incorporate fairness and algorithmic bias metrics, along with metrics for measuring accuracy and scalability.

## 7.4 Conclusion

In this chapter, we have explained our investigations regarding the disparities in the performance of VI systems that may have an effect on fairness regarding the involved stakeholders. We have used six attributes to create potentially impacted groups based on the performing artists of the query, the performing artists of the reference, and the composer of the composition, which we have identified as the relevant parties in a VI workflow. For our analyses, we collected annotations for the attributes we have considered and created the VI-Bias dataset, which we have publicly shared. As the result of 115 experiments in total, we have seen that VI systems may indeed perform differently on certain groups. Their behavior may vary depending on whether they are learning- or rule-based, or whether they use melody- or chroma-based input features, but potentially other design choices could have an impact. After presenting the obtained results, we have proposed our hypotheses on the possible causes of such performance disparities. We leave confirming or rejecting them as future work. Lastly, we have discussed the potential implications of the obtained results on multiple stakeholders involved in VI from a fairness perspective. We encourage future VI research to use the VI-Bias dataset and incorporate fairness- and algorithmic bias–related evaluation metrics along with the existing accuracy- and scalability-related ones, to get ahold of any wrongful practices.

With the experiments described in this chapter, we have concluded addressing the three main aspects that we have specified at the beginning of our research (see Section 1.3). Building upon our data-driven model MOVE (see Chapter 4), we have pursued further improvements in accuracy (see Chapter 5) and scalability (see Chapter 6) and investigated the inherent algorithmic biases of our proposed VI systems (described in this chapter). In the next chapter, we analyze the efficacy of one of our proposed systems on two industrial use cases. Although our methods described in the previous chapters have introduced substantial improvements compared with the conventional VI systems, we aim to analyze whether they meet the requirements for industrial applications.

# Chapter 8

# Applications in Industrial Use Cases

## 8.1 Introduction

Developing methods that are feasible for industrial use cases has been one of the main objectives of this dissertation (see Section 1.3). Although we have pointed our efforts toward advancing the state of the art in VI by addressing both accuracy and scalability perspectives, our evaluation methods have mostly consisted of academic datasets and performance metrics. Therefore, to have a better understanding of the state of VI systems from an industrial point of view, we teamed up with BMAT, a Barcelona-based music innovation company that focuses on music recognition and licensing applications. BMAT is also the designated partner institution for our research in the scope of the MIP-Frontiers project (see Section 1.3).

In this chapter, we introduce our attempts to evaluate one of our VI systems on two industrial use cases. Firstly, in Section 8.2, we address the problem of identifying the musical content in long recordings of live music events (i.e., concerts). There are two main challenges to this particular use case: (1) live performances of tracks tend to contain substantial differences compared to their studio-recorded counterparts, which presents a challenge in the identification process; and (2) identification systems lack the information regarding the start and end timestamps of the tracks present in such long recordings, which creates an additional challenge of obtaining proper boundaries for the list of identified musical content. Secondly, in Section 8.3, we design a large-scale retrieval experiment with a reference corpus of more than 850 k tracks. We first investigate potential gains in accuracy by using a sequential two-step system that returns a set of $K$ candidates with a fast VI system to be processed by a second VI system to retrieve the correct item(s), which leverages an idea similar to "boosting" (Schapire & Freund, 2012). We then experiment with an approximate nearest neighbor search algorithm to further reduce the retrieval time per query. We conclude the chapter with our observations on the feasibility of current VI systems in industrial applications.

## 8.2 Identifying setlists in live music recordings

### 8.2.1 Introduction

Identifying the presence of a known music track in an audio stream, ideally along with its start and end timestamps, is one of the main concerns of music recognition (or music identification) systems. While the most common and successful music recognition technologies are audio fingerprinting systems (Cano et al., 2005) that can identify recordings with slight degradations [e.g., background noise (Haitsma et al., 2001), voiceovers (Wang, 2003), or pitch shifting (Fenet et al., 2011; Joren & Leman, 2014; Sonnleitner & Widmer, 2014)], they tend to perform poorly for live performance monitoring. Live performances can incorporate many alterations from the studio recordings, including changes in tempo, key, structure, background noise, additional applause and banters, and so on. Therefore, identifying live music content typically requires VI systems, which are designed to go beyond near-exact duplicate detection (Serrà et al., 2009a; Bertin-Mahieux & Ellis, 2012; Doras & Peeters, 2019).

The setlist identification (SLI) of live music performances (i.e., full concerts) stands as a challenging branch within music recognition. In the music information retrieval community, SLI was formally defined by Wang et al. (2014) and divided into two sequential subtasks, where the first task aims to retrieve only the related metadata in the correct order, and the second task concerns further processing of the retrieved items to obtain correct timestamps. The main applications of SLI systems include automatic generation of metadata and timestamps for concerts in streaming platforms (e.g., YouTube) and copyright management for the music industry. The vast variety of music usage contexts in digital platforms makes it impossible to track music usage manually and, therefore, highlights the necessity of automatic systems.

The work by Wang et al. (2014), together with a few submissions to MIREX 2015[44] is, to the best of our knowledge, the only one specifically targeting the SLI task. Although some works in the VI literature address the use case of live performance identification (Rafii et al., 2014; Tsai et al., 2017), the proposed approaches and evaluation contexts do not consider entire concerts nor retrieving timestamps. Therefore, one cannot consider them as examples for SLI. Wang et al. (2014) assume that the artist is known for each concert. For overlapping windows, a VI system is used to return a set of candidates using thumbnails, and the final matches and their boundaries are identified among those candidates using dynamic time warping. The system is evaluated on a dataset of 20 concerts from 10 rock bands, using three metrics: edit distance, boundary deviation, and frame accuracy. Although demonstrating a plausible performance, scaling such a system to reference databases of thousands of tracks

[44]https://www.music-ir.org/mirex/wiki/2015:Set_List_Identification

stands as a difficult challenge due to the computational complexities of the used algorithms.

In this section, we study the efficacy of a set of current VI systems for SLI, considering a range of use cases related to the monitoring of live performances. To mimic a realistic industrial scenario, our approach combines the subtasks proposed by Wang et al. (2014) into a single task by creating an end-to-end workflow that takes audio signals as input and creates a final document with the retrieved metadata and timestamps. By using predetermined window and hop sizes, we compare the performances of three VI systems that produce overlapping matches, which are further processed with a novel algorithm that combines heuristic- and learning-based methods to filter out the noisy matches and to create the final results. We develop and evaluate our system using a new dataset of 75 concerts that are categorized by varying audio qualities and genres and annotated in terms of the tracks played in each concert and their timestamps. We study the impact of audio quality, genre, and reference set size (up to 56.8 k tracks) on system performance. We report our findings using four evaluation metrics following industrial practices. We publish the dataset and evaluation code at the project repository[45] (see Appendix C).

This section is based on Yesiler et al. (2021b)[46].

### 8.2.2 Methods

#### 8.2.2.1 System overview

Our workflow consists of several steps for processing a query (i.e., audio file of a concert) and a reference database to retrieve a list of tracks and their respective start and end timestamps (Figure 8.1). Firstly, we process the audio queries with a sliding window of size $W$ and a hop size $H$ (windowing is not applied to the reference tracks). For each windowed query $\mathrm{Q}_i$, we use a VI system to retrieve the most similar item from the reference database. After obtaining individual matches for each window, we perform a number of postprocessing steps to consolidate and revise those and form a final list of results. Lastly, we compute several evaluation metrics.

#### 8.2.2.2 Input representation

The first step of our system is to extract useful information from music audio signals. For this, we use cremaPCP representations (McFee & Bello, 2017), extracted with the pretrained model shared in the project repository[47] (see Section 3.5). We use a hop size of 4,096 samples for audio signals sampled at 44.1 kHz.

[45] https://github.com/furkanyesiler/setlist_id

[46] © 2021 IEEE. Reprinted, with permission, from Yesiler, F., Molina, E., Serrà, J., & Gómez, E. (2021). Investigating the efficacy of music version retrieval systems for setlist identification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 541–545.

[47] https://github.com/bmcfee/crema

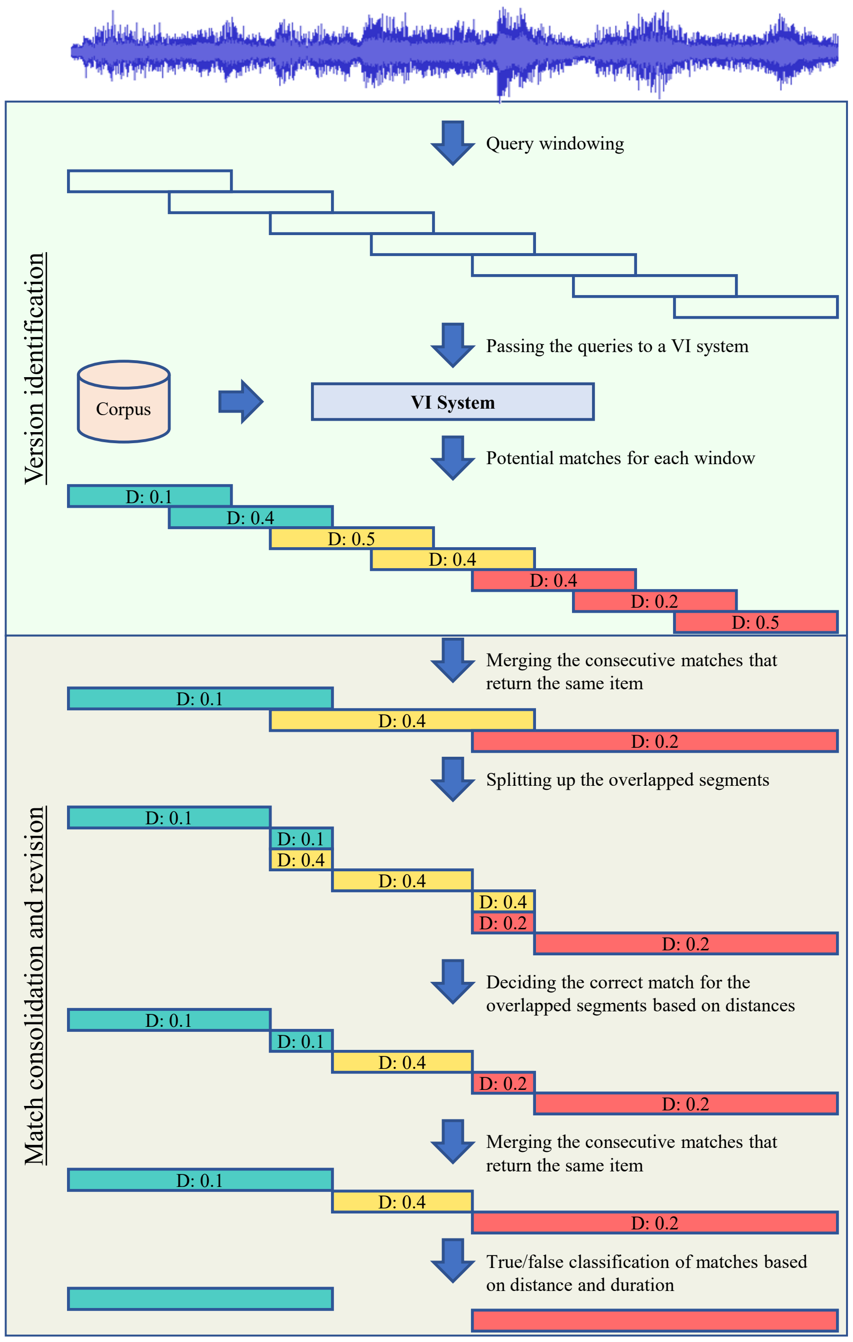

Query windowing
Version identification
Passing the queries to a VI system
Corpus
VI System
Potential matches for each window
D: 0.1
D: 0.4
D: 0.5
D: 0.4
D: 0.4
D: 0.2
D: 0.5
Merging the consecutive matches that return the same item
D: 0.1
D: 0.4
D: 0.2
Splitting up the overlapped segments
D: 0.1
D: 0.1
D: 0.4
D: 0.4
D: 0.4
D: 0.2
D: 0.2
Deciding the correct match for the overlapped segments based on distances
D: 0.1
D: 0.1
D: 0.4
D: 0.2
D: 0.2
Merging the consecutive matches that return the same item
D: 0.1
D: 0.4
D: 0.2
True/false classification of matches based on distance and duration
Match consolidation and revision

**Figure 8.1:** Overall block diagram of the proposed end-to-end workflow. Different colors in query windows indicate different tracks, D indicates the distance between that query window and the top returned track for that window.

#### 8.2.2.3 VI systems

For obtaining pairwise distance values between query windows and references, we compare three VI systems. We consider $W = \{120, 180, 240\}$ s and $H = \{15, 30, 60\}$ s. We perform initial experiments on our development set to pick the best $W$ and $H$ values for each algorithm based on the total length of correctly identified segments (see DLP metric in Section 8.2.2.6). The considered systems are:

**Re-MOVE —** Re-MOVE is a deep learning–based model that is trained with embedding distillation techniques to further improve both the accuracy and the scalability aspects of a state-of-the-art VI system (see Chapter 6). It encodes each track into an embedding vector of size 256. We transfer the pretrained weights of the model shared in the project repository[48] into an equivalent Keras (Chollet et al., 2015) model (no re-training or fine-tuning is performed). As the distance between embeddings, we use cosine distance.

**Qmax —** Qmax refers to the VI system proposed by Serrà et al. (2009a). The similarity estimation between two tracks is performed using a local alignment algorithm, and the length of the longest aligned subsequence is considered as the distance between the tracks after being normalized by the length of the reference track. We use the implementation shared in Essentia (Bogdanov et al., 2013) with the default parameters.

**2DFTM —** 2DFTM refers to the embedding-based VI system proposed by Bertin-Mahieux & Ellis (2012). It is one of the first algorithms proposed in the VI literature to address large-scale retrieval scenarios. To use beat-synchronous features as the proposed approach, we perform beat tracking on each query window and reference using onset strength envelopes precomputed with librosa (McFee et al., 2015). As the distance between embeddings, we use Euclidean distance.

#### 8.2.2.4 Consolidation and revision of potential matches

Using the VI systems described above, we compute the distances between each query window $Q_i$ and each item from the reference database. The reference track with the lowest distance to $Q_i$ is considered as its potential match. However, using a windowing scheme can create several potential matches for query segments where the processing windows are overlapped. To reduce the number of matches to a single match for any given time frame, we perform a series of operations to consolidate and revise the obtained potential matches. Note that although the VI system can be considered as the main component of our entire workflow, this last step is highly important to obtain a useful final list of results.

Our first step is to merge the consecutive overlapping matches that return the same reference track, and the distance value for the merged match is selected as the lowest distance among the respective matches. Next, to avoid the overlapping matches

[48] https://github.com/furkanyesiler/re-move

| Genre | AQ-A | AQ-B | AQ-C | Total |
|---|---|---|---|---|
| Pop/Commercial | 8 (5) | 3 | 3 | 14 (5) |
| Rock/Metal | 8 (3) | 7 | 6 | 21 (3) |
| Indie/Alternative | 5 | 7 | 3 | 15 |
| Hip-hop/Rap | 5 (2) | 0 | 3 | 8 (2) |
| Electronic | 6 | 1 | 0 | 7 |
| Total | 32 (10) | 18 | 15 | 65 (10) |

**Table 8.1:** Number of concerts per audio quality and genre. The numbers in parenthesis indicate the concerts in the development set. AQ-A contains professional recordings from large venues, AQ-B professional recordings from smaller venues, and AQ-C smartphones and video cameras.

that return different reference tracks, we obtain all possible overlaps and select the reference track that comes from the match with the lowest distance for each overlapping segment. Finally, we perform another merging step by joining any consecutive matches that return the same reference and have no gap between them (i.e., the matches that may be split in the previous step). The final results do not contain overlapping matches for any segment of the query.

Our initial experiments showed that this consolidation and revision step is useful for filtering out many incorrect matches, however, along with a few correct ones. To further reduce the number of incorrect matches, we simply train a support vector machine model for binary classification (correct/incorrect) task, using the scikit-learn library (Pedregosa et al., 2011) and the distance and duration values of correct and incorrect matches as features.

#### 8.2.2.5 Dataset

For our experiments, we collected and annotated a new dataset, ASID: automatic setlist identification dataset. It contains pre-extracted features, metadata, YouTube or Soundcloud links, and timestamp annotations for 75 concerts and all the relevant reference tracks (i.e., the tracks that are played in each concert). Concert durations range between 21.7 min and 2.5 h, with a total duration of 99.5 h. The total number of reference tracks is 1,298, with a total duration of 90.1 h. We make this dataset publicly available for the community under the Creative Commons BY-NC-SA 4.0 license[49].

ASID includes a variety of use cases regarding audio quality and genres. For this, we selected three categories for audio quality: AQ-A, AQ-B, and AQ-C. AQ-A contains high-quality recordings, mainly coming from broadcast recordings or official releases. AQ-B contains professionally recorded concerts, mainly from small venues (in general, we observe that the mixing/mastering quality for concerts in AQ-B

[49]https://creativecommons.org/licenses/by-nc-sa/4.0/

is inferior to the ones in AQ-A). Lastly, AQ-C contains smartphone or video camera recordings from varying-size venues/events. In terms of genre, we categorize the concerts into five main groups: pop/commercial, rock/metal, indie/alternative, hip-hop/rap, and electronic. The number of concerts for each audio quality and genre can be seen in Table 8.1.

We use 10 concerts (14.3 h) as a separate development set to select $W$ and $H$ for each VI system and to train a classifier for the match revision step. The references for the development set include 180 tracks. The remaining 65 concerts (85.2 h) and the related reference set are used for the main results. The total number of annotated segments for the evaluation set is 1,138, with a duration of 80.6 h.

#### 8.2.2.6 Evaluation metrics

Following common practice in industrial contexts, we evaluate our approach using: (1) true positives (TP), the number of matches (after merging and removing overlaps) that overlap the correct annotations in the ground truth (several correct matches that overlap the same annotation are counted separately); (2) false positives (FP), the number of matches (after merging and removing overlaps) that do not correspond to the related annotation; (3) detected annotations percentage (DAP), the ratio of the number of detected annotations with respect to the total number of annotations in the ground truth; and (4) detected length percentage (DLP), the ratio of the correctly identified duration of all TP with respect to the total duration of annotations in the ground truth. Moreover, we also compare the VI systems based on their required runtime to complete all the queries.

### 8.2.3 Results

#### 8.2.3.1 Overall results

Based on the DLP values obtained for the development set, we select $(W,H)$ pairs for each considered VI system. For Re-MOVE and Qmax, we select both (120,30) and (120,60) since the DLP values using those pairs result in very minor differences, and for 2DFTM, we select only (120,15). Due to the computation time needed to evaluate Qmax on our test set (see Section 8.2.3.3), we compute results for only (120,30) and simulate the results for (120,60) by skipping the matches for every second window.

The overall results for each considered system and $(W,H)$ pair show that while the performances of Re-MOVE and Qmax systems are fairly close, they both outperform 2DFTM by a considerable margin (Table 8.2). Both Re-MOVE and Qmax could identify +78% of the annotated segments (DAP metric, before the classifier), which is a considerably good performance albeit using arbitrary windows for retrieval instead of clearly segmented ones. The differences between DAP and DLP values suggest imprecise timestamp retrievals that result mainly from using fixed $W$ and $H$ values without any fine-grained refinements on the timestamp resolution.

| VI Config. | TP | FP | DAP (%) | DLP (%) |
|---|---|---|---|---|
| R - (120,30) | **936 / 771** | 1075 / 177 | **80.3 / 67.8** | 60.5 / 56.9 |
| R - (120,60) | 922 / 735 | **1013** / 112 | 79.8 / 64.6 | 60.3 / 54.8 |
| Q - (120,30) | 902 / 766 | 1217 / 132 | 78.3 / **67.8** | 60.8 / **57.5** |
| Q - (120,60) | 905 / 736 | 1039 / **75** | 78.4 / 64.9 | **60.9** / 56.0 |
| F - (120,15) | 736 / 524 | 2686 / 453 | 61.3 / 46.0 | 41.1 / 37.1 |

**Table 8.2:** Overall results for five configurations on the evaluation set. R, Q, and F denote Re-MOVE, Qmax, and 2DFTM, respectively. The left/right values denote the metrics before/after the classifier.

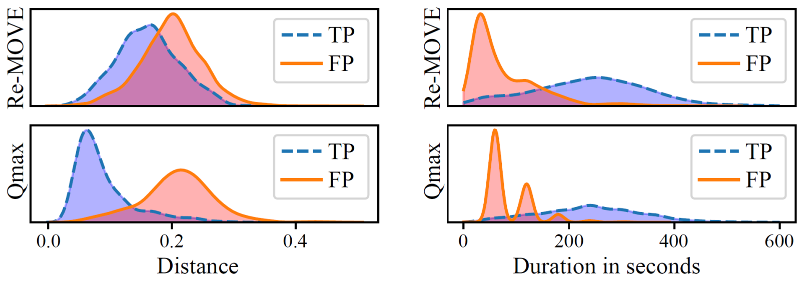


**Figure 8.2:** Distance (left) and duration (right) distributions of TP and FP for Re-MOVE - (120,30) (top) and Qmax - (120,60) (bottom).

The values before and after the classifier show that even a simple classifier can reduce the number of FPs by more than 80% while reducing the TPs by only 15–20% (excluding 2DFTM) and the DLPs by 4.5% on average. This suggests that the distance and duration distributions for TPs and FPs are different enough to enable useful classification, especially for Qmax - (120, 60) (Figure 8.2).

#### 8.2.3.2 Breakdown of results

**Audio quality —** We now present results separately for each audio quality, using a limited set of configurations (Table 8.3). The total number of annotated segments and the total duration of those (TA and TL, respectively) for categories AQ-A, AQ-B, and AQ-C are (581, 48.8 h), (241, 15.8 h), and (316, 16.9 h), respectively. The results suggest that, surprisingly, the audio quality is not the most crucial factor affecting system performance as both systems result in higher DLP values for AQ-B and AQ-C compared to AQ-A. This shows that our input representation performs robustly against noise. We hypothesize that the low performance for AQ-A may mainly result from the variety of included genres/styles.

**Genre —** The results categorized by genres can be seen in Table 8.4. TA and TL values for each category from top to bottom are (272, 17.1 h), (372, 25.6 h), (208, 12.3 h), (171, 17.6 h) and (115, 8.0 h), respectively. We observe that the performances of both

| VI Config. | TP | FP | DAP (%) | DLP (%) |
|---|---|---|---|---|
| | | *AQ-A* | | |
| R - (120, 30) | **480 / 412** | 586 / 96 | **80.6 / 71.1** | 55.6 / **53.2** |
| Q - (120, 60) | 458 / 393 | **572 / 47** | 77.5 / 67.6 | **55.5** / 52.3 |
| | | *AQ-B* | | |
| R - (120, 30) | 215 / 176 | 245 / 37 | 87.6 / 73.4 | 69.0 / 64.2 |
| Q - (120, 60) | **221 / 179** | **195 / 9** | **89.6 / 75.1** | **72.6 / 66.6** |
| | | *AQ-C* | | |
| R - (120, 30) | **241 / 183** | **244** / 44 | **74.4 / 57.6** | **67.3 / 61.1** |
| Q - (120, 60) | 226 / 164 | 272 / **19** | 71.5 / 51.9 | 65.8 / 56.8 |

**Table 8.3:** Results based on audio quality. R and Q denote Re-MOVE and Qmax, respectively. The left/right values denote the metrics before/after the classifier. AQ-A contains professional recordings from large venues, AQ-B professional recordings from smaller venues, and AQ-C smartphones and video cameras.

| VI Config. | TP | FP | DAP (%) | DLP (%) |
|---|---|---|---|---|
| | | *Pop/Commercial* | | |
| R - (120, 30) | **238 / 207** | **221** / 40 | **87.5 / 77.2** | 75.5 / **72.4** |
| Q - (120, 60) | 231 / 192 | 231 / **27** | 85.7 / 72.1 | **76.8** / 71.1 |
| | | *Rock/Metal* | | |
| R - (120, 30) | **325 / 265** | 422 / 56 | 83.3 / **70.7** | 70.1 / 66.0 |
| Q - (120, 60) | 321 / 263 | **341 / 22** | **83.9** / 70.2 | **72.4 / 66.9** |
| | | *Indie/Alternative* | | |
| R - (120, 30) | **193 / 153** | **118** / 22 | **91.8 / 73.6** | 73.3 / **67.3** |
| Q - (120, 60) | 191 / 148 | 134 / **6** | 89.4 / 71.2 | **74.0** / 66.0 |
| | | *Hip-hop/Rap* | | |
| R - (120, 30) | **76 / 60** | **246** / 47 | **43.9 / 35.1** | **19.7 / 17.8** |
| Q - (120, 60) | 63 / 47 | 264 / **16** | 36.8 / 27.5 | 15.6 / 13.6 |
| | | *Electronic* | | |
| R - (120, 30) | **104 / 86** | **68** / 12 | **87.0 / 74.8** | 68.5 / 64.9 |
| Q - (120, 60) | 99 / **86** | 69 / **4** | 85.2 / **74.8** | **69.6 / 66.4** |

**Table 8.4:** Results based on genre. R and Q denote Re-MOVE and Qmax, respectively. The left/right values denote the metrics before/after the classifier.

VI systems are consistent across genres, with "Hip-hop/Rap" being an outlier. Relying on only harmonic information for retrieval leads to a drastic performance decrease for certain musical styles (i.e., hip-hop). However, the effect of genre on system performance is not the only decisive factor. The results depicted in Figure 8.3 show that the system performance can show a large variance among concerts even within the

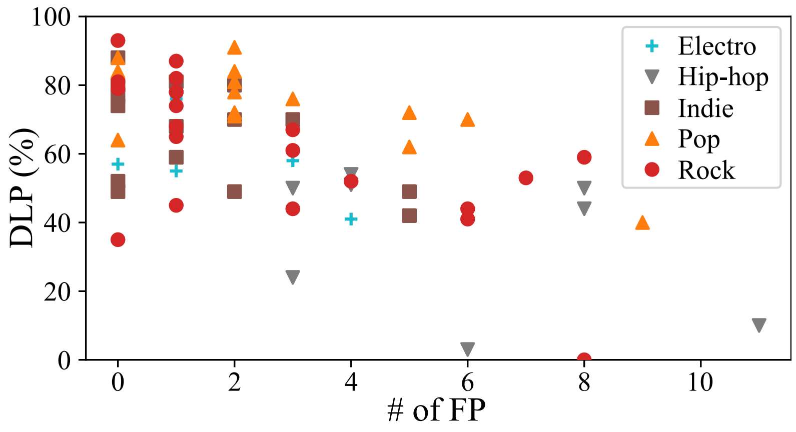


**Figure 8.3:** DLP and FP values after the classifier for each concert evaluated with Re-MOVE - (120,30), categorized by genre.

| Extra refs. | TP | FP | DAP (%) | DLP (%) |
|---|---|---|---|---|
| None | 936 / 771 | 1075 / 177 | 80.3 / 67.8 | 60.5 / 56.9 |
| 15k | 860 / 678 | 1606 / 220 | 73.6 / 59.5 | 53.0 / 48.8 |
| 30k | 836 / 661 | 1738 / 217 | 71.6 / 58.1 | 51.6 / 47.4 |
| 45k | 812 / 643 | 1785 / 241 | 69.8 / 56.6 | 49.9 / 45.9 |
| 55.7k | 812 / 639 | 1841 / 244 | 69.7 / 56.2 | 49.6 / 45.5 |

**Table 8.5:** Results of Re-MOVE - (120,30) on the MJD-expanded task.

same genre, with "Pop/Commercial" having the most consistent results.

**Reference set size —** Although we evaluate our systems for each concert using the entire reference set, a more realistic scenario should include a significantly larger reference set. For this, we gradually expand our reference set using the MTG-Jamendo dataset (MJD; Bogdanov et al., 2019), which contains the full audio tracks of 55.7 k royalty-free tracks. We assume that there is no intersection between ASID and MJD. Due to computation requirements (see Section 8.2.3.3), we only evaluate the Re-MOVE - (120,30) setting in this scenario. Table 8.5 shows that an increase in reference set size negatively affects the system accuracy. However, the system can still correctly identify 70% of the annotated segments (before the classifier), and the decrease in performance seems to be saturating after 45 k references, at least for the considered size regime.

#### 8.2.3.3 Runtime comparison

Finally, we share our observations regarding algorithm runtimes. Since computation requirements depend on $W$ and $H$, we here consider $W = 120$ and $H = 30$. Using pre-extracted cremaPCP features as input for each system and executing parallel computations with 32 cores, the Qmax algorithm takes approximately 20 days to complete the entire distance computations used for the main results in Table 8.2. Contrastingly, both Re-MOVE and 2DFTM take only 11 min. For the full MJD-expanded task in Table 8.5, the runtime of Re-MOVE only increases to 22 min (using precomputed embeddings). Although Re-MOVE and Qmax result in similar performances, the drastic difference in their runtimes suggests that Re-MOVE is the only considered system that both scales up to large-scale retrieval scenarios and achieves a plausible accuracy.

### 8.2.4 Conclusion

In this section, we have investigated the effectiveness of VI systems for automatic SLI in a wide range of use cases. For this, we have proposed an end-to-end workflow to identify the metadata and timestamps of the tracks that are present in full concerts. For the retrieval step, we have compared three VI systems in terms of accuracy and scalability. We have proposed a postprocessing algorithm that consolidates and revises the initial retrieved matches to filter out possible false positives for the final results. We have used a new dataset that contains 99.5 h of concerts, which we publicly share. Our findings suggest that while the audio quality of queries does not have a crucial effect on performance due to the robustness of our input representation against noise, the changes in musical styles/genres can have a drastic impact as our system depends solely on the harmonic information from the audio. For processing the audio queries, using predetermined window and hop sizes results in imprecise timestamps for the retrieved matches. We have also shown that increasing the size of the reference database negatively impacts system accuracy. Finally, the reported runtimes for the considered configurations show a remarkable difference between using alignment-based or embedding-based VI systems. Overall, using Re-MOVE for retrieval yields promising results towards automatic SLI in large-scale contexts; however, further improvements for the general workflow are required to address real-world live performance monitoring use cases. For further improvements in performance, two potential directions may be investigating the use of (1) an ensemble VI system that uses various musical characteristics (e.g., melody, harmony) for the retrieval phase (see Chapter 5), and (2) more elaborate false positive filtering schemes.

## 8.3 Large-scale retrieval with an industrial corpus

### 8.3.1 Introduction

Throughout this dissertation, we have designed and evaluated VI systems for retrieval settings, where we aim to retrieve the versions of a query by estimating similarities between the given query and all the tracks in a reference corpus. Although one of our major intentions is to prepare VI systems for industry-scale corpora, the largest reference set we have used so far contains roughly 56 k tracks. Considering that the size of the corpora used by modern streaming services and media platforms are in the magnitudes of millions of tracks, our previous evaluation efforts may be viewed as inadequate to infer any conclusion regarding the performance of our VI systems in industrial applications.

In this section, we present our analysis on the efficacy of one of our systems when used for large-scale corpora. We create a reference set consisting of 858,886 tracks using one of the corpora of BMAT. We first compare two VI systems in terms of their runtimes and demonstrate the drastic gap between computation time requirements of approaches that are based on local alignment and embeddings. We then design a two-step scheme where the fast, embedding-based system retrieves the top-$K$ candidates, which are later processed by the slow, local alignment–based system to produce the final list of results. We witness certain gains in accuracy using this approach, but, at the cost of increased runtimes. We also experiment with an off-the-shelf approximate nearest neighbor method to further reduce the retrieval times for the embedding-based system. With this, we obtain considerable gains in average computation time per query while having a minimal loss in accuracy.

## 8.3.2 Methods

### 8.3.2.1 Baseline systems

Our first set of experiments aim to create performance baselines using two VI systems we consider for this large-scale retrieval study. We compare the systems described below using the evaluation metrics described in Section 8.3.2.5.

**Re-MOVE —** We use the Keras version of the model (see Section 8.2.2.3) and choose cremaPCP as the input representation (see Section 3.5). Since the pairwise similarities between tracks are computed using the cosine similarity function between their embeddings, we precompute and store the embeddings for the entire reference corpus. The embeddings for the queries are computed on the fly during the retrieval phase.

**Qmax —** We use an optimized implementation of the Qmax algorithm, which is designed for reducing the computation time and made available to us by BMAT. We choose cremaPCP as input, as done for Re-MOVE. We precompute and store the cremaPCP features for the entire reference corpus since the similarity estimation with Qmax requires performing an elaborate local alignment scheme on those input features.

### 8.3.2.2 Two-step system

After setting the baselines, we investigate whether we can improve the overall accuracy by implementing a sequential two-step system. That is, we first use Re-MOVE against the entire reference corpus to retrieve the top-$K$ candidates per query. Then, we use Qmax against this selected set of $K$ candidates to obtain the final retrieval results. To study the performance with respect to the size of the set of selected candidates, we perform experiments for $K = \{5, 10, 20, 50, 100, 200, 500, 1000\}$.

In addition to using Re-MOVE and Qmax sequentially, we also experiment with a basic similarity aggregation scheme in which we refine the similarity scores between a query and each item of the selected set of $K$ candidates for that query using the similarities obtained with Re-MOVE and the two-step system (Re-MOVE then Qmax). In mathematical notation,

$$\mathcal{S}_{ij}^{\mathrm{A}} = \frac{\mathcal{S}_{ij}^{\mathrm{R}} + \mathcal{S}_{ij}^{\text{2-S}}}{2}, \tag{8.1}$$

where $\mathcal{S}_{ij}^{\mathrm{A}}$ denotes the refined similarity score between the query i and reference item j, and $\mathcal{S}_{ij}^{\mathrm{R}}$ and $\mathcal{S}_{ij}^{\text{2-S}}$ denote the similarities obtained with Re-MOVE and the two-step system, respectively. Before performing the aggregation, we make sure that the range of similarity scores obtained from both methods is bounded between -1 and 1[50].

### 8.3.2.3 Approximate nearest neighbor search

Apart from investigating potential gains in accuracy with a sequential system, we also explore potential gains in runtimes using an approximate nearest neighbor (ANN) search algorithm. When using embedding-based methods, the typical brute-force retrieval process is first to compute similarity scores between a query and all the tracks of the reference corpus, then to sort the reference items based on their similarities to the query, and lastly to retrieve the top-$K$ candidates. With an ANN algorithm, we aim to avoid similarity computation and candidate sorting steps and directly retrieve the top-$K$ candidates.

As the ANN algorithm, we use ANNOY[51], which is a publicly available C++ library with Python bindings. Although it includes several distance functions, we consider only the cosine distance. To use it, we first create an index using the embeddings of the entire reference corpus, which is represented as a forest of $n_{\text{trees}}$ trees. The documentation denotes that using more trees results in higher precision but also a larger index size. We try two indexes with $n_{\text{trees}} = \{200, 2\,\mathrm{k}\}$. After we obtain our index, we can query it with any embedding of an appropriate size. When querying, we specify the number of nodes the algorithm can inspect ($n_{\text{nodes}}$). The documentation denotes that searching more nodes returns more accurate results but requires a longer runtime. In our experiments, we try $n_{\text{nodes}} = \{20\,\mathrm{k}, 50\,\mathrm{k}, 100\,\mathrm{k}, 200\,\mathrm{k}, 500\,\mathrm{k}, 1\,\mathrm{M}, 2\,\mathrm{M}\}$.

### 8.3.2.4 Dataset

To obtain a large collection of tracks, we combine a subset of the Da-TACOS benchmark partition ($\mathrm{D}_{\mathrm{DT}}$) and a subset of one of the corpora from BMAT ($\mathrm{D}_{\mathrm{BM}}$). $\mathrm{D}_{\mathrm{DT}}$

[50]Due to the similarity estimation technique it uses, we cannot practically set a lower bound for similarity scores obtained with Qmax (therefore, the two-step system). However, we observe that scores below -1 form a very small portion of the pairwise similarities.

[51]https://github.com/spotify/annoy

includes 12,152 tracks all of which have at least one other version in the dataset. Since we have annotations only for $\mathrm{D_{DT}}$, we use this subset as the only query set.

$\mathrm{D_{BM}}$ includes 846,734 tracks with only limited metadata (e.g., track title, artist title, etc.) and without any annotations with respect to version relationships. Due to the lack of annotations, we use $\mathrm{D_{BM}}$ as a "noise" collection with which we increase the size of the reference corpus. Note that a noise collection should not contain any versions of the tracks in the query set. To ensure this, we compare the similarities between the track titles of a large corpus of over 1.5 M tracks and the track and composition titles of the query set, $\mathrm{D_{DT}}$, using the "difflib" library, which is a native Python library that uses a modified version of the gestalt pattern matching algorithm (Ratcliff & Metzener, 1988). We choose the tracks that score less than 0.6 and create $\mathrm{D_{BM}}$. The reference corpus for our experiments contain both $\mathrm{D_{DT}}$ and $\mathrm{D_{BM}}$.

#### 8.3.2.5 Evaluation metrics

We evaluate the baseline and the two-step system experiments using MAP@$K$ and MR1@$K$ (see Section 2.5.2), which are computed the same way as MAP and MR1 but consider only the first $K$ elements of the sorted results. In addition to those, we also compute the percentage of the queries for which a relevant item (a true positive) is returned at the first rank (TPP@1) and the percentage of the queries for which no relevant items are retrieved among the first $K$ candidates (NRP@$K$).

To evaluate the systems from a runtime perspective, we precompute and store the necessary representations (i.e., embeddings for Re-MOVE and cremaPCP features for Qmax) for the reference corpus. Since both Re-MOVE and Qmax use cremaPCP features in their workflows, we assume that we receive these features for each query instead of the corresponding raw audio signals. Therefore, the runtime per query is computed as (1) the time required to obtain embeddings, to compute all the similarity scores, and to sort the items based on those similarities for Re-MOVE, and (2) the time required to compute all the similarity scores and to sort the items based on those similarities for Qmax.

For the ANN experiments, we evaluate the performance by measuring the differences between the top-$K$ items returned by the ANN algorithm and by the brute-force approach, considering the brute-force results as the target. We use two metrics for this: the first one is precision at cut-off rank $K$ (Prec@$K$), which measures the ratio of the number of the items that are returned by both approaches over $K$ without considering the order of the items (e.g., if both approaches return the same items but with a different order, the score is 1); and the second one is rank accuracy at cut-off rank $K$ (RankAcc@$K$), which measures the ratio of the number of the items that are returned at the same rank by both approaches over $K$ (e.g., if both approaches return the same items but with a different order, the score is 0).

All the measured runtimes are computed with a single thread using a CPU with a clock speed of 2.60 GHz.

| $K$ | Sys. | MAP@$K$ | MR1@$K$ | TPP@1 | NRP@$K$ | $K$ | Sys. | MAP@$K$ | MR1@$K$ | TPP@1 | NRP@$K$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
|  | R | 0.225 | 2 | 0.495 | 0.301 |  | R | 0.353 | 6 | 0.495 | 0.158 |
| 5 | 2-S | 0.230 | 1 | 0.537 | N/A | 100 | 2-S | 0.378 | 4 | 0.578 | N/A |
|  | R+(2-S) | 0.231 | 1 | 0.538 | N/A |  | R+(2-S) | 0.391 | 4 | 0.584 | N/A |
|  | R | 0.321 | 2 | 0.495 | 0.259 |  | R | 0.345 | 10 | 0.495 | 0.133 |
| 10 | 2-S | 0.333 | 2 | 0.552 | N/A | 200 | 2-S | 0.371 | 7 | 0.586 | N/A |
|  | R+(2-S) | 0.334 | 2 | 0.551 | N/A |  | R+(2-S) | 0.390 | 6 | 0.592 | N/A |
|  | R | 0.361 | 2 | 0.495 | 0.224 |  | R | 0.335 | 20 | 0.495 | 0.104 |
| 20 | 2-S | 0.379 | 2 | 0.564 | N/A | 500 | 2-S | 0.362 | 13 | 0.593 | N/A |
|  | R+(2-S) | 0.382 | 2 | 0.564 | N/A |  | R+(2-S) | 0.388 | 12 | 0.601 | N/A |
|  | R | 0.359 | 4 | 0.495 | 0.184 |  | R | 0.328 | 36 | 0.495 | 0.083 |
| 50 | 2-S | 0.382 | 3 | 0.573 | N/A | 1000 | 2-S | 0.353 | 23 | 0.593 | N/A |
|  | R+(2-S) | 0.391 | 3 | 0.576 | N/A |  | R+(2-S) | 0.385 | 21 | 0.606 | N/A |

**Table 8.6:** Evaluation results for the baseline and two-step system experiments. R, 2-S, and R+(2-S) denote Re-MOVE, the two-step system, and the ensemble system that aggregates distances obtained from Re-MOVE and the two-step system, respectively.

## 8.3.3 Results

### 8.3.3.1 Baseline and the two-step system experiments

Our initial experiments with Qmax yielded a runtime of (on average) 11.04 ms per comparison. Considering the size of our reference corpus, each query is estimated to require over 2.5 h for computing all the pairwise similarities. Due to this inefficiency, we decided to use only Re-MOVE to set the baseline for the two-step experiments. Re-MOVE takes (on average) 1.2 s per query, which is several magnitudes lower than the 2.5 h per query for Qmax. The baseline performance of Re-MOVE can be seen in Table 8.6. Note that compared with its performance on smaller datasets like Da-TACOS (see Chapter 6), the MAP score is considerably lower (0.524 vs. 0.328), which confirms our doubts about results on smaller datasets not being appropriate for inferring performance on large datasets.

After obtaining the baseline with Re-MOVE, we then move to the two-step system experiments (see Table 8.6). For all the $K$ values, the two-step system outperforms Re-MOVE with relative increases varying between 2% to 8% in terms of MAP@$K$. The scores for MR1@$K$ indicate that the two-step system can locate the first relevant result at lower ranks compared to Re-MOVE. In terms of TPP@1, the relative improvements range from 8% to 20%, which shows that the two-step system is noticeably better at returning a true positive at the first rank than Re-MOVE. Lastly, we see that Re-MOVE could not find any relevant item in the top-$K$ results for 8.3% of the queries when $K = 1000$, which increases up to 30.1% for $K = 5$.

Finally, we see that a simple similarity aggregation scheme may result in considerable improvements in performance. Note that this aggregation scheme uses the pairwise similarities obtained with Re-MOVE and the two-step system, and, thus, it creates only a negligible computational load on top of the retrieval process performed for the two-step system. Compared to Re-MOVE, in terms of MAP@$K$ and TPP@1, we observe up to 9% and 2% increases in performance, respectively, depending on $K$. From

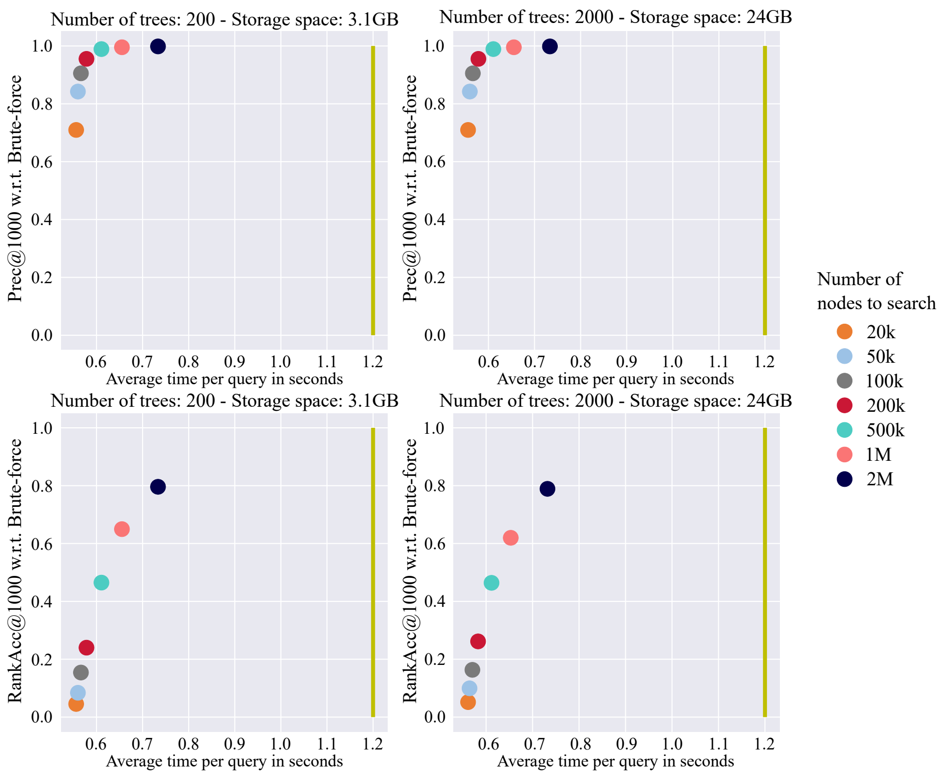


**Figure 8.4:** Evaluation results for the ANN experiments. The vertical yellow lines indicate the average time per query for the brute-force approach.

a runtime point of view, for $K = 1000$, the two-step system increases the retrieval time from 1.2 s to 12.24 s, which is a relative increase of 920%. Note that increasing the size of the reference corpus would have a decreasing effect on this difference in runtimes given that $K$ is constant. Considering that the similarity aggregation scheme provides relative increases of 17% and 22% in MAP@$K$ and TPP@1, respectively, compared to Re-MOVE, this trade-off in accuracy–scalability may seem reasonable depending on the specific use case.

### 8.3.3.2 Approximate nearest neighbor experiments

We now turn to the ANN experiments for reducing retrieval time per query. For this set of experiments, we consider only the case where $K = 1000$ as it yielded the best performance in terms of TPP@1 and NRP@$K$ while being close to its alternatives for MAP@$K$ (see Table 8.6). Figure 8.4 presents the results for the ANN experiments. The space required for storing 859 k embeddings is 3.1 GB and 24 GB for $n_{\text{trees}}$ values of 200 and 2 k, respectively. In terms of average runtime per query, we observe improvements for all the $n_{\text{trees}}$ and $n_{\text{nodes}}$ combinations, with values ranging from 0.56 s to 0.73 s. However, for the cases where $n_{\text{nodes}}$ values are below 100 k, Prec@$K$ scores stay under 0.9 and reach around 1 only for $n_{\text{nodes}} \geq 500\,\text{k}$. RankAcc@$K$ results, on the other hand, show that for a score above 0.8, $n_{\text{nodes}}$ should be at least 2 M. The discrepancies between Prec@$K$ and RankAcc@$K$ results suggest that the ANN algorithm is useful for retrieving the same set of documents in a much faster way; however, the order of the documents may show variations.

### 8.3.4 Conclusion

In this section, we have shared the results of a large-scale retrieval study using a reference corpus with more than 858 k tracks. We have shown that local alignment–based systems, even with an optimized version for fast retrieval, are not suitable for such large-scale corpora. Our embedding-based system, on the other hand, has demonstrated computation times appropriate for such use cases. However, in terms of identification performance, our system has resulted in a worse performance compared with the evaluation scores shared in previous chapters, mainly due to performing against such a large-scale corpus. To improve the system performance, we have proposed a two-step system, where the embedding-based model retrieves a set of top-$K$ candidates, and the local alignment–based method processes only that selected set of candidates. With such a sequential system, we have demonstrated clear gains in performance, however, at the cost of longer retrieval times per query. Finally, we have experimented with an ANN algorithm to further reduce the retrieval time for our embedding-based system.

## 8.4 Chapter conclusion

In this chapter, we have evaluated one of our VI systems in terms of identification performance and runtime on two industrial use cases. Since the beginning of this dissertation, we have touched on the industry aspect of our research, which arises out of mainly two points: (1) the, perhaps, most important application scenario of automatic VI systems is digital rights management, which is primarily an industrial application, and (2) the MIP-Frontiers project, which includes our research, has an emphasis on promoting collaboration between academia and industry. By collaborating with BMAT, we have identified and worked on two main use cases. Firstly, we have proposed an end-to-end workflow to identify setlists of live concerts, which incorporates a VI system for identification and a heuristic-based algorithm for revising and eliminating false positives to obtain a clean final report. We have tested our workflow on concerts of a variety of audio qualities and from different genres. Our results have suggested that although our VI system, among the systems we considered, is the most appropriate one for such a task in terms of accuracy and scalability, the current performance may not be sufficient for using it in systems that interact with clients. Secondly, we have set up a large-scale version retrieval study where we have evaluated our VI system using a reference corpus of more than 858 k tracks. After obtaining the evaluation scores for our system, we have proposed a sequential two-step system to improve performance. We have observed certain increases in the accuracy, however, along with certain decreases in the runtime efficiency. We have also experimented with an approximate nearest neighbor search algorithm to further improve the retrieval time of our embedding-based VI system. Overall, our findings for both use cases suggest that although drastic improvements in retrieval speed have been made in VI systems compared with the conventional local alignment–based ones, the current level of identification performance may not be sufficient for deploying these systems as-is. To bridge the gap between the goals of academic research and the requirements of industrial applications, further research dedicated to addressing such a gap is essential.

# Chapter 9
# Conclusion

## 9.1 Summary

We have begun this dissertation by motivating the reader on the vital role of musical versions in the world's musical heritage. By doing that, we have aimed to shed light on the fact that unlike many musical characteristics that may vary depending on the musical tradition at hand, the intuition and skills for identifying musical versions are rooted deep in certain cognitive skills that are crucial for our survival as species. While being able to identify versions brings unequivocal value to our relationship with music, efforts toward modeling such a skill to integrate it into automatic computational systems may further advance our capabilities for creating, listening, and appreciating music. Although one may lose sight of such motivations while building automatic version identification systems, we believe that delving into such philosophical aspects of scientific research may help one to gain perspective toward their ultimate goals.

Following the focus of the MIP-Frontiers project on further encouraging collaboration between the academic and industrial institutions, we have set the main objective of this dissertation to be the development of automatic version identification systems that can be used for industry-scale corpora. Toward that objective, we have specified four major ideas to focus on. Firstly, we have decided to concentrate our efforts on designing data-driven systems since they had been proven effective in other MIR applications from both performance and computation speed perspectives. However, instead of embracing a fully task-agnostic, data-driven approach, we have investigated ways of incorporating task-specific domain knowledge into the data-driven models we have developed. The results have demonstrated the effectiveness of that strategy. Secondly, for a system to be approved for deployment in industrial use cases, it is required to provide reliable results. Therefore, we have explored ideas to improve the accuracy of data-driven VI systems. By exploiting feature fusion and ensemble approaches that process different musical characteristics, we have achieved substantial improvements in evaluation scores. Thirdly, the fact that the new content in music corpora increases rapidly forces many retrieval systems to produce decisions as fast as possible. To improve the retrieval speed of VI systems, we have explored and proposed embedding distillation techniques through which we have demonstrated improvements in the scalability of existing VI systems without experiencing a decrease in accuracy. Lastly, to be aware of any harmful effects VI systems may have toward certain groups of musicians, we have investigated the algorithmic bias of our VI systems. Since our main objective is to use such systems in industrial use cases, preventing any unfair outcomes that may arise as a result of that should be of utmost importance for a sustainable music ecosystem. Apart from the four major ideas outlined above, we have also tested the effectiveness of our systems on two industrial use cases in a collaboration with a broadcast monitoring and licensing company. The results suggest that our systems have made substantial progress in terms of being both accurate and scalable; however, further research with the collaboration of academic and industrial institutions seems to be required to deploy such systems for industrial applications.

improvements and optimizations seem to be required to deploy such systems for industrial applications.

We now provide an overview of our contributions and key results. With this, we aim to concisely summarize the main takeaways that have been provided at the end of each chapter. Finally, we conclude the dissertation by providing our views on open issues, challenges, and future directions in VI research.

## 9.2 Contributions and key results

We now present an overview of the contributions of this dissertation. Complementary to the summary that we have provided in Section 1.4, we here include a brief review of the key results we have presented in each chapter. In addition to the overall contributions outlined below, Appendix B contains the breakdown of specific contributions of the author for each chapter.

**C.1 A discussion of the concepts of "musical version" and "musical work" from MIR, musicological, and legal perspectives —** In Chapter 1, we have argued that in MIR, the musical version and musical work concepts are defined based on the practical reasons such as how datasets are annotated. Although such definitions do not correspond to how these concepts are defined in musicological and legal studies, finding a universal definition is also a difficult task due to the complexity of the problem. Therefore, by discussing these concepts from different contexts, we have aimed to increase awareness of the richness of perspectives that exist when carrying out research on the topic of musical versions.

**C.2 A survey of the key ideas, techniques, datasets, and evaluation methods that were proposed throughout two decades of VI research —** In Chapter 2, we have provided a detailed survey of the VI research from the 2000s up to the present. By using an existing taxonomy, we have classified the individual components of the VI systems. To the best of our knowledge, our survey is the only one that includes both the knowledge- and data-driven modules designed for VI. We have also categorized a number of ideas that had been proposed to improve the accuracy and scalability of VI systems into six main categories. Lastly, we have reviewed the publicly available datasets for VI, including the recent ones that consist of tens of thousands of tracks for data-driven VI research.

**C.3 A large-scale quantitative analysis on the frequency and extent of changes in musical characteristics between version pairs —** In Chapter 3, we have shared the results of the first large-scale quantitative analysis for a better understanding of the frequency and extent of changes in musical characteristics when musicians perform versions. We have focused on five main characteristics: key, tempo, timing, structure, and semantic aspects. We have devised custom measures for estimating pairwise similarities in timing, structure, and semantic aspects. Our analysis has shown that all the inspected characteristics show variations between versions on different levels. Based on those results, we have argued that VI systems must incorporate strategies to handle such variations for better performance.

**C.4 A data-driven VI system that incorporates explicit, musically motivated modules for handling the common changes in musical characteristics between versions —** In Chapter 4, we have proposed a novel, deep learning–based VI system that combines knowledge-driven design decisions with a data-driven workflow. Our system, MOVE, has been designed to encode each track into a fixed-size embedding vector, regardless of the track duration. With this, computationally expensive similarity estimation functions have been reduced to computing simple and fast functions like Euclidean distance. Based on the findings from Chapter 3, we have used explicit modules for handling transposition and structure changes that are common between versions. We have also designed a data augmentation strategy that simulates changes in key, tempo, and timing for each track, with the goal of making the proposed model more robust against such changes. We have followed a metric learning approach to train our network due to its advantages for dealing with a large number of classes. The resulting model, MOVE, have achieved state-of-the-art performance by improving upon existing systems from accuracy and scalability perspectives.

**C.5 A data-driven fusion approach to VI for achieving accuracy improvements by combining the information from systems that process different musical characteristics —** In Chapter 5, we have studied possible means of further improving the performance of MOVE by proposing a data-driven fusion approach. Although achieving a plausible performance, MOVE extracts and processes only harmonic information from each track, which is suboptimal due to the complexity of connections between versions. To address this, we have experimented with feature fusion and ensemble approaches that combine two systems processing melodic and harmonic features. Along with a simple distance averaging scheme, we have proposed a data-driven approach for creating a fusion system, which has drastically improved the performance of the individual systems that it consists of and has resulted in state-of-the-art performances on two publicly available datasets.

**C.6 An investigation toward methods for speeding up the retrieval phase of VI systems without compromising the performance —** In Chapter 6, we have investigated a number of existing and proposed embedding distillation techniques to reduce the embedding size of data-driven VI systems while maintaining their accuracy. While training MOVE, we have observed that larger embedding sizes have yielded a better performance, however, at the cost of larger storage and longer runtime requirements. With embedding distillation techniques, we have shown that learning a more compact embedding space with fewer dimensions while preserving identification performance is possible when the feature extractor part of the networks incorporates stronger priors rather than randomly initialized weights. We have obtained 99% smaller embeddings that yielded up to a 3% performance increase. We have called the resulting model Re-MOVE, which stands for "reduced MOVE."

**C.7 An investigation into the inherent algorithmic biases of VI systems —** In Chapter 7, we have conducted the first study ever to investigate the algorithmic bias in VI systems. Acknowledging the impact that VI systems in industrial applications may have on musicians, we have performed a large set of experiments by considering five existing VI systems of different characteristics, six relevant attributes (year, popularity, country, language, year, prevalence), and three stakeholders (performing artists of the query and the reference tracks, and the composer). We have designed our experiments with the purpose of evaluating whether any of the VI systems favor certain groups of musicians or composers based on the attributes we have considered. Our findings have shown that VI systems may indeed demonstrate discrepancies in identification performance based on their characteristics (e.g., using harmony- or melody-based features, being learning- or heuristic-based, etc.), and certain characteristics of the performing artists and composers. We have also shared our hypotheses on the reasons for observed performance discrepancies for a number of cases.

**C.8 Studies on the efficacy of our systems for industrial use cases —** In Chapter 8, we have evaluated one of our systems, Re-MOVE, on two industrial use cases: setlist identification of live music events and large-scale version retrieval. Both of these studies have been conducted in collaboration with BMAT.

In Section 8.2, we have proposed an end-to-end workflow for identifying the musical content and respective timestamps in long recordings of live music events. After identifying the tracks in overlapped analysis windows using a VI system, our workflow further processes the initial candidates for consolidating and revising them using a heuristic approach. We have also employed a simple classifier for reducing the number of false positives in the final list of results. We have compared three VI systems, including Re-MOVE, while keeping the rest of the workflow the same. We have performed our experiments on concerts of varying audio qualities and genres. Our results have shown that among the ones we have considered, Re-MOVE was the only VI system that has reached a promising performance while providing fast retrieval; however, further improvements are needed to deploy such a system in industrial applications.

In Section 8.3, we have performed a large-scale retrieval study using a reference corpus of more than 858 k tracks. After setting a baseline with Re-MOVE, we have proposed a sequential two-step system, where Re-MOVE estimates similarities between a query and the entire reference corpus and returns a set of candidates, and a second system processes the same query and only the selected candidates to refine the results. We have shown that such a sequential system brings slight improvements in performance but considerably increases the retrieval time. We have also investigated potential gains in retrieval speed using Re-MOVE and an approximate nearest neighbor algorithm. We have demonstrated a substantial decrease in runtime for the retrieval phase compared with a brute-force approach.

**C.9 A critical discussion on the current open issues and challenges of VI systems along with potential future directions —** In Section 9.3 (see below), we outline some current open issues, challenges, and future directions in VI research. Apart from the issues like the accuracy–scalability trade-off that has been discussed elsewhere and many researchers are aware of, we discuss a number of issues that previous research had not addressed thoroughly. We categorize such issues and future directions into five main groups: (1) the task definition, (2) evaluation methodologies, (3) trade-offs regarding scalability, (4) accuracy gaps for underrepresented content, and (5) applications of VI systems. By pointing out these issues and potential directions, we hope to contribute to better development practices in the field and increased diversity on the topics VI research is involved with.

In addition to the contributions and key results described above, the research presented in this dissertation resulted in three publicly available datasets, five code repositories, a website, a system demo, and a tutorial at an international conference. The details of these additional contributions can be found in Appendix C.

## 9.3 Open issues, challenges, and future directions

Although VI research has made substantial progress in the last 20 years, there are many challenges that have yet to be addressed. We now outline some of such challenges and current open issues to provide guidance for researchers that are interested in contributing to the field. These challenges include, but are not limited to, (1) the task definition itself, (2) evaluation methodologies, (3) trade-offs that arise when scaling VI systems, (4) accuracy gaps on certain underrepresented types of content, and (5) the variety of applications and the different treatments they require.

### 9.3.1 Task definition

As discussed in Section 1.2.2, the definition of a musical version, especially a “cover song,” may differ in various contexts. To avoid such differences, we have so far favored a quite permissive definition in this dissertation, labeling all the tracks that are derived from a musical composition as versions. Although this definition is convenient for introducing and discussing VI from an academic research point of view, applications that consider legal aspects of VI (e.g., detecting copyright infringements) may require different definitions that are more suitable for their purposes.

The definition of a version for legal applications often needs to be based upon the rightsholders (typically songwriters and recording labels) rather than the musical connections. For example, a composer may copyright a new arrangement of a folk song that is in the public domain. According to our inclusive definition of a version, the arrangement would be a version of the original, but legally, it may be a separate entity. These rights themselves are often not well-defined, and there are a number of famous lawsuits[52] about whether or not the creators of a track must pay royalties to the rightsholders. Publishing data, which provides a legal link between track metadata and composition metadata, often exists in text form, linking songwriters/composers, track titles, and recording/composition identifiers such as ISRCs/ISWCs[53].

There are cases where this metadata is the only information separating two nearly identical tracks (such as an original release and a remastered release) into different legal entities. Thus, purely audio-based VI for legal applications is not possible in many cases, and any successful system must consider additional information such as editorial metadata to disambiguate unclear cases.

Apart from the legal perspective, current VI systems are typically built around a particular notion of which musical features are important for determining whether two tracks are versions of one another. These notions fit well for some music traditions, such as Western pop and classical music; however, there are other musical traditions that break some of these assumptions. For example, in Indian art music, certain melodic phrases or motifs are crucial for identifying the ragas (roughly speaking, modes) that tracks belong to (Gulati et al., 2016). From a Western point of view, two tracks that include versions of the same melodic phrase would mostly mean that they originate from the same composition; however, in Indian art music, it can simply mean that they belong to the same raga.

### 9.3.2 Evaluation methodologies

Evaluation of VI systems is typically performed on well-curated datasets (see Section 2.5.1) using mostly rank-aware evaluation metrics (see Section 2.5.2). These datasets are dominated by music with singing voice, music with well-defined versions, music from mostly pop, rock, and jazz genres, and generally do not contain duplicates. Industry-scale music corpora, however, often have quite different distributions. In such corpora, near-exact duplicates are very common, it is difficult to

[52] https://www.bbc.com/culture/article/20190605-nine-most-notorious-copyright-cases-in-music-history

[53] https://isrc.ifpi.org/

apply the concept of versions for the majority of music, genres like rap and electronic music constitute a large proportion of the data, plus a non-negligible subset of the tracks does not contain singing voice. This presents a challenge when extrapolating the performance of a system on an industry-scale corpus using evaluations performed on well-curated datasets. Furthermore, there are several limitations of the commonly used evaluation metrics. Firstly, they are highly sensitive to the number of relevant tracks in the corpus per query, which may not be appropriate for VI since some compositions may have hundreds of versions while some others may have only a few, or even zero. Additionally, the metrics mostly consider only the rank ordering of the tracks and not the distances between the query and the retrieved items, which makes it difficult to assess how well-separated the relevant items are from the irrelevant ones.

Near-duplicates (i.e., content that is the same but distorted enough so that a standard music fingerprinting algorithm will not provide a match) present a particular challenge: given a query, VI systems will naturally assign a lower distance to a near-duplicate compared to other versions that may contain several changes in musical characteristics. This highlights both of the previously mentioned issues regarding the evaluation metrics. For example, for precision at cut-off rank $K$ (Prec@$K$; see Section 2.5.2), the presence of duplicates can both increase and decrease the metric in an unpredictable way, as this changes the number of relevant tracks for a given query. Consider this toy example: corpus A has no duplicates, and corpus B is an extended version of corpus A, where each item has one near-duplicate. Now consider a VI system that, for a given query, would return 5 relevant results out of the first 10 for corpus A (i.e., Prec@$10 = 0.5$). If the relevant results are in positions 1 through 5, the equivalent query for corpus B would have 10 relevant results (i.e., Prec@$10 = 1.0$). Conversely, if the relevant results are in positions 6 through 10 (for corpus A), the equivalent query for corpus B would have 0 relevant results (i.e., Prec@$10 = 0.0$).

Music without well-defined versions, such as ambient music and soundscapes, leave an open question: how should VI systems handle this type of content? Practically, a VI system should not retrieve any matches when no versions are present. However, this kind of content typically does not have clear melodic, harmonic, or structural characteristics. As a result, the features VI has historically used are typically close to zero everywhere and are confidently, and incorrectly, clustered together as a tight group by VI systems.

The genre distributions of research datasets may introduce a bias toward certain input representations and musical characteristics. For example, the success of PCP representations in VI is fairly easy to explain for datasets having many tracks from the pop, rock, and jazz genres. However, the performances of state-of-the-art systems on other genres where rhythmic and timbral properties are highlighted is an underexplored issue in VI (see Section 8.2). Considering the popularity of hip-hop and electronic music genres (and their subgenres), this is clearly an issue to be addressed in VI research to be useful in the current music ecosystem.

Finally, VI research, except for a subfield focusing on Western classical music, has underexplored how systems behave for instrumental music, largely due to the existing datasets being dominated by music with singing voice. As a result, the performance of current VI systems on instrumental music is not well-understood, and thus the performance on industry-scale corpora, which contain a considerable percentage of instrumental content, cannot be inferred.

### 9.3.3 Scalability trade-offs

Although VI has clear industrial applications in the current music ecosystem, the scope of scalability-related discussions in research papers is rather limited. It has been demonstrated that vector-based techniques provide large benefits in computation and memory requirements compared with alignment-based ones, but no systematic evaluation of vector-based techniques has been performed from a scalability perspective. The scalability considerations in such works are typically limited to the size of the embedding vectors. However, the computations that produce these vectors (e.g., feature extraction algorithms, or deep neural network layers) are mostly ignored. Therefore, here, we highlight two directions to cover this underexplored perspective.

Firstly, for the vector-based techniques, in particular, there is an additional accuracy–scalability trade-off that arises especially in industry-scale datasets: retrieving the $k$-nearest neighbors for a query. Consider a vector-based system, with vectors in $\mathbb{R}^d$ under the Euclidean norm. In order to retrieve the $k$-nearest results from a corpus of $N$ items in an exact manner, at least $O(N \log N)$ operations are required, and these exact algorithms are difficult to improve due to the "curse of dimensionality." Even if pruning techniques are employed to reduce $N$, when performing large numbers of lookups for retrieving $k$-nearest neighbors, the computational efficiency can greatly influence the speed and cost of deploying a VI system. In practice, approximate nearest neighbor search algorithms[54] are used (see Section 8.3). These algorithms provide efficient approximations for finding the $k$-nearest neighbors for high-dimensional data over large corpora and introduce a trade-off between the time per query and the recall (i.e., the percentage of true nearest neighbors returned): the higher the recall, the slower the query. The degree of trade-off varies by algorithm and dataset, but roughly, at 80% recall, these algorithms provide a speedup of a factor between 100 and 1000 over an exact computation. For a fixed speedup, the achieved recall typically decreases as the dimensionality of the vectors increases, which highlights an additional motivation for the dimensionality reduction (see Chapter 6) or data projection methods (see Section 2.3.5).

Secondly, to better understand the time and memory complexities of VI systems, additional metrics such as floating point operations per second (FLOPS) and peak memory usage can be reported. The goal then would be to use such metrics to compare entire workflows when performing matching for many queries against a large corpus, from

[54]For example, https://github.com/erikbern/ann-benchmarks

feature extraction up through the final similarity estimation steps. Although not all VI research needs to aim for industrial-level scalability, such information can be useful for comparing application-oriented VI systems.

### 9.3.4 Accuracy gaps

Improving the accuracy of VI systems has been the main goal of most VI research, and we now highlight a number of ideas to accelerate the progress toward this goal. The commonly used features described in Section 2.3.1 have been successful at capturing relevant information for quantifying similarities between versions for most mainstream music. However, in some edge cases (e.g., cross-genre versions, *a cappella* versions, and versions of drum solos), such features may fail drastically. Therefore, to further improve the accuracy of current systems, other musical dimensions should be considered. The biggest challenge here is to find ways to exploit these uncommon musical characteristics while keeping in mind the principal invariances a VI system must consider. For example, for certain cases, a particular rhythmic pattern may facilitate identification, but using only rhythmic information for identifying versions would certainly fail in a large body of popular music where the rhythmic patterns are similar for various compositions.

A musical dimension that has not been considered in previous works until very recently is the lyrics. Lyrics could be a major factor for improving VI accuracy for versions that share the same lyrics/rhyme patterns but little else, such as those with no prominent melodic or harmonic characteristics to rely on, those with greatly varying singing styles, or those with lyrics as the only connecting property. Estimating lyrics from directly polyphonic audio is a challenging task that has been explored, but with limited success. However, as has been done for harmony and melody, features capturing approximate lyric information (e.g., phoneme-related features of the singing voice) could be a useful, complementary signal to the existing feature set. Additionally, onset and rhythm information is not yet well-captured by the existing features, yet they are key features for determining similarity between certain types of music, such as rap and electronic music.

Track metadata, such as tags describing the high-level musical properties, can also be a powerful, underexplored signal for VI (Correya et al., 2018). Typically, systems aim to be invariant to such high-level characteristics, but using tags such as “vocal” or “instrumental” as a way of conditioning may improve system performances, as they could inform the systems about what kind of properties they should focus on exploiting.

As for more extreme examples, there are categories of music where none of the existing or aforementioned dimensions adequately capture what makes tracks similar or not, such as sound art, soundscapes (e.g., rain sounds, city streets), ambient music (e.g., singing bowls, drones), etc. In these cases, the composition is closely tied to the properties of the recording itself, such as the precise sound qualities and placement of events in time. To address such cases, music fingerprinting techniques could be applied, e.g., as a preprocessing step of a VI system.

Another potential direction for improving accuracy is to fully embrace data-driven representation learning. While hand-designing features has proven to be useful for VI, it introduces a bias toward what VI systems are able to model. Alternatively, end-to-end learning paradigms can be explored, where given a sufficient amount of data, a system learns which properties of the track are most important for the task. In particular, these techniques give systems the potential to uncover relations beyond melody/harmony that are relevant for identifying matching versions. These techniques have seen some success in other related domains such as speech recognition and have not yet been explored for VI.

Finally, there is an opportunity for VI systems to place more focus on postprocessing operations, such as version set enhancement methods. Such operations are generally computationally cheap and are proven to improve accuracy. However, except for a few research papers in the early 2010s, this research direction has been on standby.

### 9.3.5 Emphasis on subfields and applications

Most VI research focuses on the general problem of identifying and retrieving versions of tracks, but there are a number of underexplored subfields and practical applications of these systems. Certain subfields of VI focus on particular types of versions in order to address or exploit specific characteristics and challenges. For example, in versions of Western classical music (Zalkow & Müller, 2020; see the column "Performances" in Figure 1.2), the musical variations are typically limited to those in tempo, timing, key, "noise," and possibly structure (e.g., the presence or absence of repeats). Therefore, the systems designed to be used in such cases focus more on being robust to noise and timing distortions while assuming melodic and harmonic characteristics are likely to be shared between versions.

Rather than identifying full tracks, there is the interesting subproblem of identifying versions of short queries, or phrases (e.g., 3–15 seconds; Müller et al., 2005). Considering the difficulties that music fingerprinting systems have with identifying versions (even live performances), such an application scenario could address a particular need for end-users. Moreover, given the ability to identify versions of short phrases, their origins could be identified, which could enable musicologists to create phylogenetic trees of musical phrases. Using these, musical influences within and between musical genres and styles can be studied to have a better understanding of the evolution of musical practice.

Another less-explored application is in "setlist identification," wherein the task is to identify versions from a long recording consisting of a sequence of versions of different tracks (see Section 8.2). The long recording could be, for example, a live recording of a concert, a DJ set, or a medley. Such long recordings are usually processed with overlapping windows that span typically 1–2 minutes. However, this is error-prone, as the windows may cross multiple tracks. To avoid this, a segmentation step can be performed beforehand, but such algorithms may also introduce erroneous segments, especially when live tracks are interrupted briefly for banter or applause. Therefore, solving problems other than identification performance may be crucial for a VI system to be used for setlist identification.

In some applications, the typical setup of having a fixed corpus does not hold, and instead, the VI problem exists in an "online" setting. In this case, the goal is to build a graph of connections over time (e.g., as new tracks are added to a corpus) in an online fashion. In this application, there are many open problems, such as how to avoid error propagation (e.g., by applying version set enhancement methods), and exploring efficient ways to perform the online steps.

Finally, applications that match audio to editorial metadata have not been widely explored. A common example is matching a registered "composition," which exists purely as text metadata, to a corpus of tracks. In this context, once there is at least one track connected to a composition, standard VI techniques can be employed. Further, it is common to match new tracks to existing "compositions" which already have several matching tracks, moving the problem from track-to-track matching to track-to-clique matching.

Version identification research has come a long way in the last 20 years: from comparing symbolic sequences to data-driven representation learning, a great number of techniques and ideas have been studied to address this task that is deeply connected to the history of musical practice. The quest for developing the perfect VI system may never come to an end as the limitless creativity of musicians is most likely to triumph no matter how well VI systems perform at their jobs. However, we shall remind ourselves that this is not a zero-sum game: musicians getting more creative with their versions leads to richer experiences for music listeners and more interesting directions for VI research. The road so far has paved the way for a promising future for the next generation of VI systems, but there is still a long way to go, because, all in all, *"there is nothing that says a great song cannot be interpreted at any time in any way."*[55]

[55]Phil Ramone, as cited by Plasketes (2005).

M. Furkan Yesiler, Barcelona, November 18, 2021.

# Bibliography

The numbers at the end of each bibliographic entry indicate the pages in which it is cited.

Adams, H., Chepushtanova, S., Emerson, T., Hanson, E., Kirby, M., Motta, F., Neville, R., Peterson, C., Shipman, P., & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, *18*(1), 218–252. 52.

Avram, H. (2011). Cover & remix: Paradigms of adaptation in installation art. *Cultural Legitimation*, *2*(1), 6–30. 8, 9.

Barrett, F. S., Grimm, K. J., Robins, R. W., Wildschut, T., Sedikides, C., & Janata, P. (2010). Music-evoked nostalgia: Affect, memory, and personality. *Emotion*, *10*(3), 390–403. 2.

Barron, A. (2006). Copyright law's musical work. *Social & Legal Studies*, *15*(1), 101–127. 11, 12.

Barsalou, L. W. (1991). Deriving categories to achieve goals. *Psychology of Learning and Motivation*, *27*, 1–64. 2.

Baxter, G. & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. *Interacting with Computers*, *23*(1), 4–17. 97.

Bello, J. P. (2007). Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 239–244. 27.

Bello, J. P. & Pickens, J. (2005). A robust mid-level representation for harmonic content in music signals. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 304–311. 19, 34.

Bertin-Mahieux, T. & Ellis, D. P. W. (2012). Large-scale cover song recognition using the 2D Fourier transform magnitude. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 241–246. 22, 27, 29, 30, 31, 33, 54, 56, 68, 81, 94, 112, 115.

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., & Lamere, P. (2011). The million song dataset. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 591–596. 40, 41, 85.

Böck, S., Korzeniowski, F., Schlüter, J., Krebs, F., & Widmer, G. (2016). Madmom: A new Python audio and music signal processing library. In *Proc. of the ACM International Conference on Multimedia*, pp. 1174–1178. 47.

Böck, S., Krebs, F., & Widmer, G. (2015). Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 625–631. 49.

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., Roma, G., Salamon, J., Zapata, J. R., & Serra, X. (2013). ESSENTIA: An audio analysis library for music information retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 493–498. 47, 99, 115.

Bogdanov, D., Won, M., Tovstogan, P., Porter, A., & Serra, X. (2019). The MTG-Jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery (ML4MD) Workshop, International Conference on Machine Learning (ICML)*. 120.

Brauneis, R. (2014). Musical work copyright for the era of digital sound technology: Looking beyond composition and performance. *Tulane Journal of Technology and Intellectual Property*, *17*. 11, 12.

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 535–541. 88.

Butler, J. (2010). Musical works, cover versions and 'Strange Little Girls'. *Volume !*, *7*(1), 42–72. 11.

Cai, K., Yang, D., & Chen, X. (2016). Two-layer large-scale cover song identification system based on music structure segmentation. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6. 22, 31, 37.

Cano, P., Batlle, E., Haitsma, J., & Kalker, T. (2005). A review of audio fingerprinting. *Journal of VLSI Signal Processing*, *41*(3), 271–284. 20, 112.

Carroll, M. W. (2003). A primer on U.S. intellectual property rights applicable to music information retrieval systems. *University of Illinois Journal of Law Technology & Policy*, *2003*(2), 313–328. 12.

Carroll, M. W. (2005). The struggle for music copyright. *Florida Law Review*, *57*, 907–961. 12.

Casey, M. A. & Slaney, M. (2006). Song intersection by approximate nearest neighbor search. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 144–149. 22, 31, 38.

Chen, N., Li, W., & Xiao, H. (2018a). Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, *77*(2), 2629–2652. 21, 27, 30, 31, 33, 36, 37, 54, 55, 56, 67, 68, 81, 94.

Chen, Y., Wang, N., & Zhang, Z. (2018b). DarkRank: Accelerating deep metric learning via cross sample similarities transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2852–2859. 89.

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396. 47, 52.

Chollet, F. et al. (2015). Keras. https://keras.io. 115.

Chouldechova, A. & Roth, A. (2020). A snapshot of the frontiers of fairness in machine learning. *Communications of the ACM*, *63*. 98.

Constandinides, C. (2019). “You just got covered”: YouTube cover song videos as examples of para-adaptation. In W. Schäfke-Zell & J. Fehrle (Eds.) *Adaptation in the Age of Media Convergence*, pp. 111–132. Amsterdam University Press. 9.

Cooper, B. L. (2018). Cover me: The stories behind the greatest cover songs of all time. *Popular Music and Society*, *41*(2), 216–219. 8, 9.

Correya, A., Hennequin, R., & Arcos, M. (2018). Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features. *arXiv preprint arXiv:1808.10351*. 37, 137.

Cusic, D. (2005). In defense of cover songs. *Popular Music and Society*, *28*(2), 171–177. 8, 9.

Davies, D. L. & Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-1*(2), 224–227. 89.

Day, B. (2011). In defense of copyright: Record labels, creativity, and the future of music. *Seton Hall Journal of Sports and Entertainment Law*, *21*(1), 61–103. 12.

Degani, A., Dalai, M., Leonardi, R., & Migliorati, P. (2013). A heuristic for distance fusion in cover song identification. In *Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pp. 1–4. 36.

Dineley, S. (2014). *Covers uncovered: A history of the "cover version," from Bing Crosby to the Flaming Lips*. Master's thesis, The University of Western Ontario, London, Ontario, Canada. 9.

Dodd, J. (2014). Upholding standards: A realist ontology of standard form jazz. *The Journal of Aesthetics and Art Criticism*, *72*(3), 277–290. 11.

Doras, G. (2020). *Cover detection using deep learning*. Ph.D. thesis, Sorbonne Université, France. 19, 21.

Doras, G. & Peeters, G. (2019). Cover detection using dominant melody embeddings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 107–114. 23, 26, 28, 29, 30, 32, 34, 39, 40, 42, 75, 85, 99, 102, 103, 112.

Doras, G., Yesiler, F., Serrà, J., Gómez, E., & Peeters, G. (2020). Combining musical features for cover detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 279–286. 15, 24, 26, 27, 36, 74, 160.

Du, X., Yu, Z., Zhu, B., Chen, X., & Ma, Z. (2021). ByteCover: Cover song identification via multi-loss training. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 551–555. 24.

Edelsbrunner, H. & Harer, J. (2010). *Computational topology: An introduction*. American Mathematical Society. 51.

Elezi, I., Vascon, S., Torcinovich, A., Pelillo, M., & Leal-Taixé, L. (2020). The group loss for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 277–294. 92.

Ellis, D. P. W. (2007). The "covers80" cover song data set. http://labrosa.ee.columbia.edu/projects/coversongs/covers80/. 40, 41.

Ellis, D. P. W. & Poliner, G. E. (2007). Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. IV, pp. 1429–1432. 20, 27, 28, 30, 33.

Fenet, S., Richard, G., & Grenier, Y. (2011). A scalable audio fingerprint method with robustness to pitch-shifting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 121–126. 112.

Ferraro, A., Serra, X., & Bauer, C. (2021). Break the loop: Gender imbalance in music recommenders. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval (CHIIR)*, pp. 249–254. 98.

Fisher, J. A. (2018). Jazz and musical works: Hypnotized by the wrong model. *The Journal of Aesthetics and Art Criticism*, *76*(2), 151–162. 11.

Fletcher, R. R., Nakeshimana, A., & Olubeko, O. (2021). Addressing fairness, bias, and appropriate use of artificial intelligence and machine learning in global health. *Frontiers in Artificial Intelligence*, *3*. 98.

Foote, J. (2000). ARTHUR: Retrieving orchestral music by long-term structure. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 6, 19, 33.

Foster, P., Dixon, S., & Klapuri, A. (2015). Identifying cover songs using information-theoretic measures of similarity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(6), 993–1005. 34.

Foucard, R., Durrieu, J.-L., Lagrange, M., & Richard, G. (2010). Multimodal similarity between musical streams for cover version detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5514–5517. 21, 36.

Frankle, J. & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 88.

Goehr, L. (1994). *The Imaginary Museum of Musical Works: An Essay in the Philosophy of Music*. Oxford, UK: Clarendon Press. 11.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2004). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 513–520. 91.

Gómez, E. (2006). Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, *18*(3), 294–304. 48.

Gómez, E. & Herrera, P. (2006). The song remains the same: Identifying versions of the same piece using tonal descriptors. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 180–185. 28, 30, 33.

Gómez, E., Ong, B., & Herrera, P. (2006). Automatic tonal analysis from music summaries for version identification. In *Audio Engineering Society (AES) Convention 121*. 20, 27, 31, 33.

Gracyk, T. (2013). Covers and communicative intentions. *The Journal of Music and Meaning*, *11*, 22–46. 8, 9.

Grosche, P., Müller, M., & Serrà, J. (2012). Audio content-based music retrieval. In M. Müller, M. Goto, & M. Schedl (Eds.) *Multimodal Music Processing*, *Dagstuhl Follow-Ups*, vol. 3, pp. 157–174. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. XVII, 5, 159.

Gulati, S., Serrà, J., Ishwar, V., & Serra, X. (2016). Discovering rāga motifs by characterizing communities in networks of melodic patterns. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 286–290. 134.

Haitsma, J., Kalker, T., & Oostveen, J. (2001). Robust audio hashing for content identification. In *International Workshop on Content-Based Multimedia Indexing (CBMI)*. 112.

Han, J., Zhao, T., & Zhang, C. (2019). Deep distillation metric learning. In *Proceedings of the ACM Multimedia Asia (MMAsia)*. 89.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1135–1143. 88.

Hanson, S. J. & Pratt, L. Y. (1988). Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 177–185. 88.

Hermans, A., Beyer, L., & Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint arXiv: 1703.07737*. 64.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 88.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*(2), 65–70. 103.

Humphrey, E. J., Nieto, O., & Bello, J. P. (2013). Data driven and discriminative projections for large-scale cover song identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 4–9. 23, 27, 29, 33, 34.

Huron, D. (2006). *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, Massachusetts: The MIT Press. 2.

Jiang, C., Yang, D., & Chen, X. (2020). Learn a robust representation for cover song identification via aggregating local and global music temporal context. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. 28, 30, 32, 34.

Joren, S. & Leman, M. (2014). Panako - A scalable acoustic fingerprinting system handling time-scale and pitch modification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 259–264. 112.

Khadkevich, M. & Omologo, M. (2013). Large-scale cover song identification using chord profiles. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 233–238. 27, 28, 38.

Korzeniowski, F. & Widmer, G. (2016). Feature learning for chord recognition: The deep chroma extractor. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 37–43. 55.

Krebs, F., Böck, S., & Widmer, G. (2015). An efficient state-space model for joint tempo and meter tracking. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 72–78. 50.

Kurth, F. & Müller, M. (2008). Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(2), 382–395. 22, 34, 38.

LeCun, Y., Denker, J. S., & Solla, S. A. (1989). Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 598–605. 88.

Lee, J., Chang, S., Choe, S. K., & Lee, K. (2018). Cover song identification using song-to-song cross-similarity matrix with convolutional neural network. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 396–400. 34.

Levinson, J. (1980). What a musical work is. *The Journal of Philosophy*, *77*(1), 5–28. 11.

Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. In *IEEE International Conference on Data Mining (ICDM)*, pp. 911–916. 89.

Magnus, C., Magnus, P. D., & Uidhir, C. M. (2013). Judging covers. *The Journal of Aesthetics and Art Criticism*, *71*(4), 361–370. 8.

Marolt, M. (2008). A mid-level representation for melody-based retrieval in audio collections. *IEEE Transactions on Multimedia*, *10*(8), 1617–1625. 26, 33, 38.

Martin, B., Brown, D. G., Hanna, P., & Ferraro, P. (2012). BLAST for audio sequences alignment: A fast scalable cover identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 529–534. 22, 38.

Massey, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, *46*(253), 68–78. 48, 103.

McFee, B. & Bello, J. P. (2017). Structured training for large-vocabulary chord recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 188–194. 47, 49, 55, 113.

McFee, B. & Ellis, D. P. W. (2014). Analyzing song structure with spectral clustering. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 405–410. Taipei, Taiwan. 49.

McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the Python in science conference (SciPy)*, pp. 18–25. 47, 115.

McFee, B., Salamon, J., & Bello, J. P. (2018). Adaptive pooling operators for weakly labeled sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(11), 2180–2193. 61.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, *54*(6). 97.

Meseguer-Brocal, G., Cohen-Hadria, A., & Peeters, G. (2018). DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 431–437. 89.

Meyer, L. B. (1956). *Emotion and Meaning in Music*. Chicago, USA: The University of Chicago Press. 2.

Mosser, K. (2008). Cover songs: Ambiguity, multivalence, polysemy. *Philosophy Faculty Publications*. 8.

Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., & Singh, S. (2017). No fuss distance metric learning using proxies. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 360–368. 91.

Müller, M., Kurth, F., & Clausen, M. (2005). Audio matching via chroma-based statistical features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 288–295. 19, 27, 31, 33, 138.

Olteanu, A., Garcia-Gathright, J., de Rijke, M., Ekstrand, M. D., Roegiest, A., Lipani, A., Beutel, A., Olteanu, A., Lucic, A., Stoica, A.-A., Das, A., Biega, A., Voorn, B., Hauff, C., Spina, D., Lewis, D., Oard, D. W., Yilmaz, E., Hasibi, F., Kazai, G., McDonald, G., Haned, H., Ounis, I., van der Linden, I., Garcia-Gathright, J., Baan, J., Lau, K. N., Balog, K., de Rijke, M., Sayed, M., Panteli, M., Sanderson, M., Lease, M., Ekstrand, M. D., Lahoti, P., & Kamishima, T. (2021). FACTS-IR: Fairness, accountability, confidentiality, transparency, and safety in information retrieval. *ACM SIGIR Forum*, *53*(2), 20–43. 97.

Ortega, J. L. (2021). Cover versions as an impact indicator in popular music: A quantitative network analysis. *PLoS ONE*, *16*(4), e0250212. 8.

Osmalskyj, J. (2017). *A Combining Approach to Cover Song Identification*. Ph.D. thesis, University of Liege, Belgium. 45.

Osmalskyj, J., Van Droogenbroeck, M., & Embrechts, J.-J. (2016). Enhancing cover song identification with hierarchical rank aggregation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 136–142. 21, 22, 36, 37.

Park, W., Kim, D., Lu, Y., & Cho, M. (2019). Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3967–3976. 89.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. 88, 116.

Plasketes, G. (2005). Re-flections on the cover age: A collage of continuous coverage in popular music. *Popular Music and Society*, *28*(2), 137–161. 1, 8, 9, 139.

Porcaro, L. & Gómez, E. (2019). 20 years of playlists: A statistical analysis on popularity and diversity. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 130–136. 98.

Rafii, Z., Coover, B., & Han, J. (2014). An audio fingerprinting system for live version identification using image processing techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 644–648. 112.

Ratcliff, J. W. & Metzener, D. (1988). Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, *13*(7), 46–51. 124.

Ravuri, S. & Ellis, D. P. W. (2010). Cover song detection: From high scores to general classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 65–68. 21, 36.

Räz, T. (2021). Group fairness: Independence revisited. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, pp. 129–137. 98, 99.

Reuter, M., Wolter, F.-E., & Peinecke, N. (2006). Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design*, *38*(4), 342–366. 49.

Robinson, K., Brown, D., & Schedl, M. (2020). User insights on diversity in music recommendation lists. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 446–453. 98.

Rosch, E. (1999). Reclaiming concepts. *Journal of Consciousness Studies*, *2*(11–12), 61–77. 2.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., & Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, *8*(3), 382–439. 2.

Rouse, D., Watkins, A., Porter, D., Harer, J., Bendich, P., Strawn, N., Munch, E., DeSena, J., Clarke, J., Gilbert, J., Chin, S., & Newman, A. (2015). Feature-aided multiple hypothesis tracking using topological and statistical behavior classifiers. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIV*, pp. 189–200. 51.

Sailer, C. & Dressler, K. (2006). Finding cover songs by melodic similarity. *Music Information Retrieval Evaluation eXchange (MIREX)*. 28.

Salamon, J., Serrà, J., & Gómez, E. (2012). Melody, bass line, and harmony representations for music version identification. In *Proceedings of the International World Wide Web Conference (WWW): International Workshop on Advances in Music Information Research (AdMIRe)*, pp. 887–894. 21, 26, 36.

Schapire, R. E. & Freund, Y. (2012). *Boosting: Foundations and Algorithms*. Cambridge, Massachusetts: The MIT Press. 111.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 815–823. 64.

Seetharaman, P. & Rafii, Z. (2017). Cover song identification with 2D Fourier transform sequences. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 616–620. 65, 69.

Selbst, A. D., Boyd, D., Friedler, S. A., Venkatasubramanian, S., & Vertesi, J. (2019). Fairness and abstraction in sociotechnical systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 59–68. 97.

Serrà, J. (2011). *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions*. Ph.D. thesis, Universitat Pompeu Fabra, Spain. XVII, 1, 7, 45, 68, 73.

Serrà, J., Gómez, E., & Herrera, P. (2010). Audio cover song identification and similarity: background, approaches, evaluation, and beyond. In Z. W. Ras & A. A. Wieczorkowska (Eds.) *Advances in Music Information Retrieval, Studies in Computational Intelligence*, vol. 16, chap. 14, pp. 307–332. Berlin, Germany: Springer. 6.

Serrà, J., Gómez, E., Herrera, P., & Serra, X. (2008). Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(6), 1138–1151. 20, 27, 29, 30.

Serrà, J., Kantz, H., Serra, X., & Andrzejak, R. G. (2012). Predictability of music descriptor time series and its application to cover song detection. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(2), 514–525. 34.

Serrà, J., Pascual, S., & Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. In *Artificial Intelligence Research and Development*, *Frontiers in Artificial Intelligence and Applications*, vol. 308, pp. 120–129. Amsterdam, The Netherlands: IOS Press. 61.

Serrà, J., Serra, X., & Andrzejak, R. G. (2009a). Cross recurrence quantification for cover song identification. *New Journal of Physics*, *11*, 093017. 21, 27, 29, 30, 31, 33, 54, 56, 68, 81, 94, 99, 112, 115.

Serrà, J., Zanin, M., Laurier, C., & Sordo, M. (2009b). Unsupervised detection of cover song sets: Accuracy improvement and original identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 225–230. 21, 35, 54, 56, 67, 68, 81, 94.

Shakespeare, D., Porcaro, L., Gómez, E., & Castillo, C. (2020). Exploring artist gender bias in music recommendation. In *Proceedings of the ImpactRS Workshop at ACM Recommender Systems*. 98.

Silva, D. F., Chin-Chia, M. Y., Batista, G. E. A. P. A., & Keogh, E. J. (2016). SiMPle: Assessing music similarity using subsequences joins. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 23–29. 54, 56, 65, 68, 69, 81, 94, 159.

Silva, D. F., de Souza, V. M. A., & Batista, G. E. A. P. A. (2015). Music shapelets for fast cover song recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 441–447. 22, 40, 41.

Silva, D. F., Falcão, F. V., & Andrade, N. (2018). Summarizing and comparing music data and its application on cover song identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 732–739. 31, 55, 69.

Smith, T. F. & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, *147*(1), 195–197. 30.

Sonnleitner, R. & Widmer, G. (2014). Quad-based audio fingerprinting robust to time and frequency scaling. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pp. 1–8. 112.

Swartz, A. (2002). MusicBrainz: A semantic web service. *IEEE Intelligent Systems*, *17*(1), 76–77. 100.

Tagg, P. (2000). 'The Work': An evaluative charge. In M. Talbot (Ed.) *The Musical Work: Reality or Invention?*, pp. 153–167. Trowbridge: Liverpool University Press. 11.

Tralie, C. J. (2017). Early MFCC and HPCP fusion for robust cover song identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 294–301. 21, 27, 33, 35, 36, 37.

Tralie, C. J. & Bendich, P. (2015). Cover song identification with timbral shape sequences. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 38–44. 27, 35, 54, 56.

Tralie, C. J. & McFee, B. (2019). Enhanced hierarchical music structure annotations via feature level similarity fusion. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 201–205. 49.

Tsai, T., Prätzlich, T., & Müller, M. (2017). Known-artist live song identification using audio hashprints. *IEEE Transactions on Multimedia*, *19*(7), 1569–1582. 112.

Tversky, B. & Hemenway, K. (1984). Objects, parts, and categories. *Journal of Experimental Psychology*, *113*(2), 169–193. 1, 2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6000–6010. 32.

Wang, A. (2003). An industrial-strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. 112.

Wang, J.-C., Yen, M.-C., Yang, Y.-H., & Wang, H.-M. (2014). Automatic set list identification and song segmentation for full-length concert videos. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 239–244. 112, 113.

Wright, L. (2019). Ranger - a synergistic optimizer. https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer. 92.

Wu, C.-W. & Lerch, A. (2017). Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 613–620. 89.

Xu, X., Chen, X., & Yang, D. (2018). Key-invariant convolutional neural network toward efficient cover song identification. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. 23, 29, 32, 34, 39, 40, 41, 59, 65, 69.

Ye, Z., Choi, J., & Friedland, G. (2019). Supervised deep hashing for highly efficient cover song detection. In *Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 234–239. 30.

Yesiler, F., Doras, G., Bittner, R. M., Tralie, C. J., & Serrà, J. (2021a). Audio-based musical version identification: Elements and challenges. *IEEE Signal Processing Magazine*, *38*(6), 115–136. XIX, 14, 17, 159, 160, 163.

Yesiler, F., Miron, M., Serrà, J., & Gómez, E. (2022). Assessing algorithmic biases for musical version identification. In *ACM International Conference on Web Search and Data Mining (WSDM)*. (to appear). 16, 99, 160, 163.

Yesiler, F., Molina, E., Serrà, J., & Gómez, E. (2021b). Investigating the efficacy of music version retrieval systems for setlist identification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 541–545. 16, 113, 160, 163.

Yesiler, F., Serrà, J., & Gómez, E. (2020a). Accurate and scalable version identification using musically-motivated embeddings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 21–25. 15, 23, 29, 32, 34, 39, 58, 159, 162.

Yesiler, F., Serrà, J., & Gómez, E. (2020b). Less is more: Faster and better music version identification with embedding distillation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 884–892. 16, 86, 160, 163.

Yesiler, F., Tralie, C., Correya, A. A., Silva, D. F., Tovstogan, P., Gómez, E., & Serra, X. (2019). Da-TACOS: A dataset for cover song identification and understanding. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 327–334. 15, 27, 37, 40, 41, 42, 46, 159, 162.

Yu, L., Yazici, V. O., Liu, X., van de Weijer, J., Cheng, Y., & Ramisa, A. (2019a). Learning metrics from teachers: Compact networks for image embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2907–2916. 89.

Yu, Z., Xu, X., Chen, X., & Yang, D. (2019b). Temporal pyramid pooling convolutional neural network for cover song identification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4846–4852. 23, 28, 32, 34, 39, 40, 42, 65, 69, 85.

Zalkow, F. & Müller, M. (2020). Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music. *Applied Sciences*, *10*(1). 24, 31, 85, 138.

Zbikowski, L. M. (2002). *Conceptualizing Music: Cognitive Structure, Theory, and Analysis*. Oxford, UK: Oxford University Press. 1.

Zhai, A. & Wu, H. (2019). Classification is a strong baseline for deep metric learning. In *Proceedings of the British Machine Vision Conference (BMVC)*. 91.

# Appendix A
# Publications by the Author

## Peer-reviewed journals

- **Yesiler, F.**, Doras, G., Bittner, R. M., Tralie, C. J., & Serrà, J. (2021). Audio-based musical version identification: Elements and challenges. *IEEE Signal Processing Magazine*, 38(6), 115–136.

## Full articles in peer-reviewed conferences

- **Yesiler, F.**, Miron, M., Serrà, J., & Gómez, E. (2022). Assessing algorithmic biases for musical version identification. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. (to appear).

- **Yesiler, F.**, Molina, E., Serrà, J., & Gómez, E. (2021). Investigating the efficacy of music version retrieval systems for setlist identification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 541–545.

- **Yesiler, F.**, Serrà, J., & Gómez, E. (2020). Less is more: Faster and better music version identification with embedding distillation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 884–892.

- Doras, G., **Yesiler, F.**, Serrà, J., Gómez, E., & Peeters, G. (2020). Combining musical features for cover detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 279–286.

- **Yesiler, F.**, Serrà, J., & Gómez, E. (2020). Accurate and scalable version identification using musically-motivated embeddings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 21–25.

- **Yesiler, F.**, Tralie, C., Correya, A. A., Silva, D. F., Tovstogan, P., Gómez, E., & Serra, X. (2019). Da-TACOS: A dataset for cover song identification and understanding. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 327–334.

- **Yesiler, F.**, Bozkurt, B., & Serra, X. (2018). Makam recognition using extended pitch distribution features and multi-layer perceptrons. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pp. 249–253.

- **Yesiler, F.**, & Ramirez, R. (2018). A machine learning approach to classification of phonation modes in singing. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pp. 362–367.

## Theses

- **Yesiler, F.** (2018). Analysis and automatic classification of phonation modes in singing. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain.

Additional and up-to-date information about the author may be found at the author's web page[56]

[56]https://furkanyesiler.github.io/

Appendix B

# Breakdown of Contributions by the Author

Below is the list of specific contributions of the author (F.Y.) for each of the chapters in this dissertation:

**Chapter 1 —** Parts of this chapter are based on Yesiler et al. (2021a), which was led by F.Y. Specifically, F.Y. and Rachel Bittner (R.B.; Spotify, France) prepared Figure 1.2. F.Y. drafted and finalized all the written content in this chapter that is based on Yesiler et al. (2021a). Moreover, Figure 1.1 is taken from Grosche et al. (2012), which is licensed under Creative Commons BY-ND.

**Chapter 2 —** This chapter is based on Yesiler et al. (2021a), which was led by F.Y. Specifically, F.Y., R.B., Guillaume Doras (G.D.; IRCAM, France), and Christopher Tralie (C.T.; Ursinus College, USA) wrote the initial draft. G.D. and F.Y. prepared Figures 2.1 and 2.2. F.Y., G.D., R.B., C.T., and Joan Serrà (J.S.; Dolby Laboratories, Spain) contributed to the final version of the article.

**Chapter 3 —** This chapter is based on Yesiler et al. (2019), which was led by F.Y. Specifically, F.Y., C.T., Albin Correya (A.C.; Moodagent, Denmark), Diego F. Silva (D.S.; Federal University of São Carlos, Brazil), and Philip Tovstogan (P.T.; Universitat Pompeu Fabra, Spain) wrote the initial draft. F.Y. curated the dataset, organized the metadata, and extracted the shared features. C.T. developed the techniques for structure, timing, and semantic aspect analysis outlined in 3.3 and prepared Figures 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, and 3.7. A.C. and C.T. led the implementation of existing algorithms outlined in Section 3.4. D.S. implemented the algorithm SiMPle (Silva et al., 2016). F.Y., C.T., A.C., D.S., and P.T. contributed to the final version of the paper. Emilia Gómez (E.G.; Universitat Pompeu Fabra, Spain) and Xavier Serra (X.S.; Universitat Pompeu Fabra, Spain) supervised the project.

**Chapter 4 —** This chapter is based on Yesiler et al. (2020a), which was led by F.Y. Specifically, F.Y. designed and performed the experiments. F.Y. and J.S. contributed to the final version of the paper. J.S. and E.G. supervised the project.

**Chapter 5 —** This chapter is based on Doras et al. (2020), which was led by G.D. Specifically, G.D. and F.Y. formulated the research problem and the experimental design. G.D. and F.Y. performed the feature extraction for the datasets SHS4- and Da-TACOS, respectively. G.D. and F.Y. designed the neural network model MICE. G.D. designed and implemented the distance averaging scheme. G.D. and F.Y. designed the data-driven late fusion scheme, and G.D. performed the experiments. G.D. and F.Y. analyzed and discussed the results. G.D. prepared Figure 5.2. G.D. wrote the initial draft of the paper. G.D., F.Y., J.S., E.G., and Geoffroy Peeters (G.P.; TelecomParis, France) contributed to the final version of the paper. J.S., E.G., and G.P. supervised the project. F.Y. performed the error analysis in Section 5.3.5, using the pairs of tracks provided by G.D.

**Chapter 6 —** This chapter is based on Yesiler et al. (2020b), which was led by F.Y. Specifically, F.Y. designed and performed the experiments. F.Y. and J.S. contributed to the final version of the paper. J.S. and E.G. supervised the project.

**Chapter 7 —** This chapter is based on Yesiler et al. (2022), which was led by F.Y. Specifically, F.Y. designed and performed the experiments. F.Y., J.S., and Marius Miron (M.M.; Universitat Pompeu Fabra, Spain) discussed the experimental results. M.M. contributed to the fairness and bias aspects of the paper. F.Y. and J.S. contributed to the final version of the paper. M.M., J.S., and E.G. supervised the project.

**Chapter 8 —** Section 8.2 of this chapter is based on Yesiler et al. (2021b), which was led by F.Y. Specifically, F.Y. designed and performed the experiments. F.Y. and Emilio Molina (E.M.; BMAT, Spain) contributed to the dataset design and curation. F.Y. and E.M. discussed the experimental results. F.Y., E.M., and J.S. contributed to the final version of the paper. E.M., J.S., and E.G. supervised the project. Section 8.3 of this chapter is based on a work F.Y. did under the supervision of E.M. at BMAT. Specifically, F.Y. and E.M. designed the experiments. F.Y. and Guillem Cortès (G.C.; Universitat Pompeu Fabra, Spain) prepared the dataset. F.Y. performed the experiments and analyzed the results.

**Chapter 9 —** Section 9.3 of this chapter is based on Yesiler et al. (2021a), which was led by F.Y. Specifically, R.B. and F.Y. wrote the initial draft. F.Y., R.B., G.D., C.T., and J.S. contributed to the final version of the article.

# Appendix C
# Additional Contributions

## Datasets

**Da-TACOS —** To address the need for a large dataset designed for training and evaluating VI systems, we collected and publicly shared Da-TACOS under Creative Commons BY-NC-SA 4.0 license. It includes training, validation, and benchmark partitions with 83,904 tracks (14,499 cliques), 14,000 tracks (3,500 cliques), and 15,000 tracks (3,000 cliques), respectively. All the aforementioned partitions include disjoint sets of musical works to avoid evaluation bias. The version annotations are obtained using the API of SHS, and they are shared under the Creative Commons BY-NC 3.0 license. Due to legal constraints, we share pre-extracted features rather than audio files. All audio files used for computing the features are encoded in MP3 format with varying bit rates, and their sample rate is 44.1 kHz. Along with the features, we share the metadata that is linked to the composition and performance IDs used in SHS. The list of pre-extracted features, details about the included metadata, and the instructions on how to obtain the dataset can be found at the supplementary website of the project[57].

Figure A.1 shows the distributions of tracks per clique for the training (83.9 k tracks, left) and the validation partitions (14.0 k tracks, right). The number of tracks per clique ranges from 2 to 109 in the training set, and from 2 to 11 in the validation set. Our intention was to mimic real-world data where it is more likely to have more unique works with less number of versions, rather than having a balanced dataset in terms of the number of tracks per clique. In the benchmark partition, however, we aimed for a more balanced distribution in terms of the number of tracks per clique. This partition includes 13,000 tracks in 1,000 cliques (13 tracks for each clique), and 2,000 noise tracks that do not belong to any other clique. By having more tracks per clique, we aimed to increase the chances of including challenging cases that may confuse the VI systems.

[57] https://mtg.github.io/da-tacos/

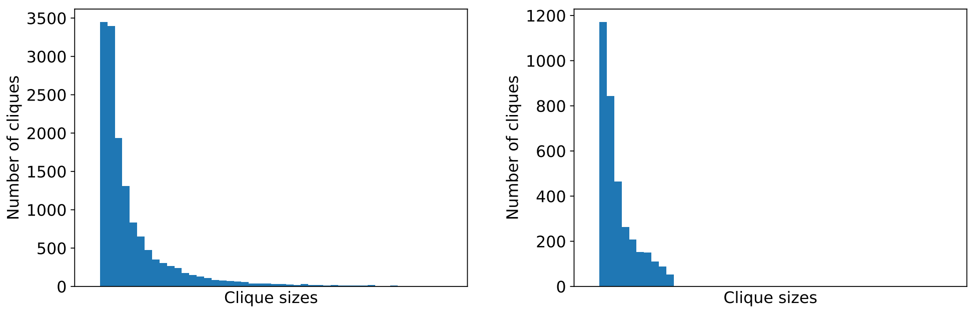


**Figure A.1:** Distribution of the number of tracks per clique for the training (left) and the validation (right) partitions.

**VI-Bias —** For analyzing the algorithmic bias in VI systems, we created the VI-Bias dataset. We used a subset of the SHS4- dataset (see Section 2.5.2) and annotated the tracks in terms of their metadata and label categories we used in our experiments (gender, popularity, country, language, year, prevalence). We share the VI-Bias dataset publicly under Creative Commons BY-NC-SA 4.0 license. More information on the dataset can be found in Section 7.2.3.1. The instructions on how to obtain the dataset can be found at the project repository[58].

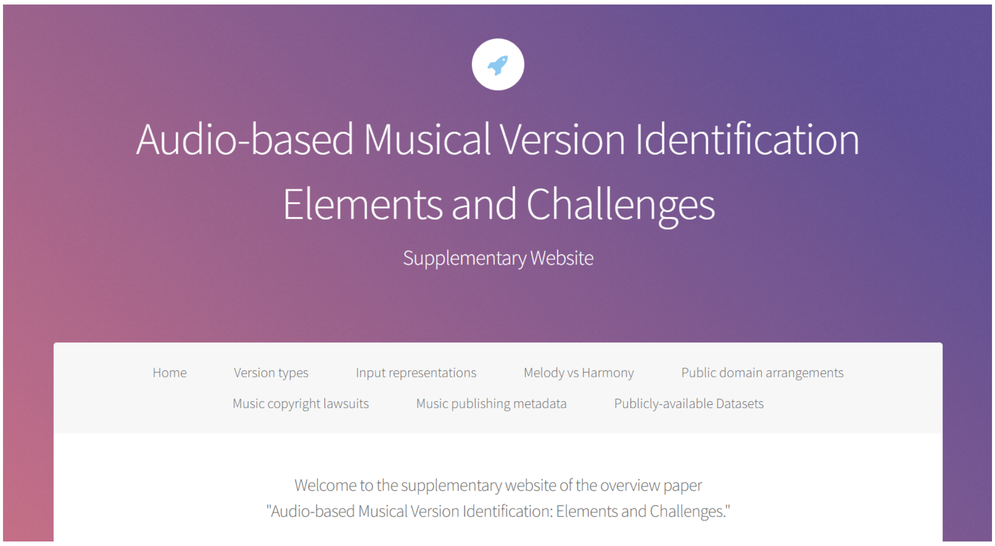
**ASID —** For developing and evaluating our setlist identification system, we created the ASID dataset. It contains 75 concerts and all the tracks that are played in those concerts. Due to legal constraints, we share pre-extracted features, metadata, YouTube or Soundcloud links for the concerts and individual tracks under Creative Commons BY-NC-SA 4.0 license. More information on the dataset can be found in Section 8.2.2.5. The instructions on how to obtain the dataset can be found at the project repository[59].

## Code repositories

**acoss** (https://github.com/furkanyesiler/acoss) — This repository contains the code for reproducing the feature extraction for preparing the Da-TACOS dataset and the benchmarking experiments described in Yesiler et al. (2019). The feature extraction part uses open-source libraries from the MIR community. The benchmarking part contains implementations of seven knowledge-driven VI systems for facilitating reproducible evaluation in VI research.

**move** (https://github.com/furkanyesiler/move) — This repository contains the code for reproducing the model training and evaluation described in Yesiler et al. (2020a). Along with detailed explanations for how to use the code, the repository contains the pretrained weights for the MOVE model (see Chapter 4).

[58] https://github.com/furkanyesiler/vi_bias
[59] https://github.com/furkanyesiler/setlist_id



Welcome to the supplementary website of the overview paper "Audio-based Musical Version Identification: Elements and Challenges."

**Figure A.2:** The landing page of the supplementary website for Yesiler et al. (2021a).

**re-move** (https://github.com/furkanyesiler/re-move) — This repository contains the code for reproducing the model training and evaluation described in Yesiler et al. (2020b). It contains the detailed explanations for how to use the code and the pre-trained weights for the Re-MOVE model (see Chapter 6).

**vi_bias** (https://github.com/furkanyesiler/vi_bias) — This repository contains the code for reproducing the algorithmic bias evaluation experiments described in Yesiler et al. (2022), and instructions on how to obtain the VI-Bias dataset.

**setlist_id** (https://github.com/furkanyesiler/setlist_id) — This repository contains the code for reproducing the setlist identification experiments described in Yesiler et al. (2021b), and instructions on how to obtain the ASID dataset.

## Other

**Website on versions** (https://furkanyesiler.github.io/musical_version_id_spm/) — We created a supplementary website to facilitate understanding some of the concepts mentioned in Yesiler et al. (2021a). The landing page can be seen in Figure A.2. The content in the website can be categorized into seven main groups:

- **Version types:** This part contains a number of version pairs that are challenging for the VI systems to identify. Moreover, we include 50 different versions of the track "Total Eclipse of the Heart" by Bonnie Tyler, annotated by the types of versions.

- **Input representations:** This part contains four types of input representations that VI systems use, extracted for the track “Don’t Stop Believin’” by Journey.

- **Melody vs. Harmony:** This part contains a number of version and non-version pairs that would cause contradicting outcomes for VI systems that process melodic information and harmonic information.
- **Public domain arrangements:** This part contains two examples of public domain arrangements that are protected by copyright.
- **Music copyright lawsuits:** This part contains tracks from a number of famous music copyright lawsuits.
- **Music publishing metadata:** This part contains an example of the music publishing metadata and how unorganized it can be.
- **Publicly available datasets:** This part contains a list of publicly available datasets for VI.

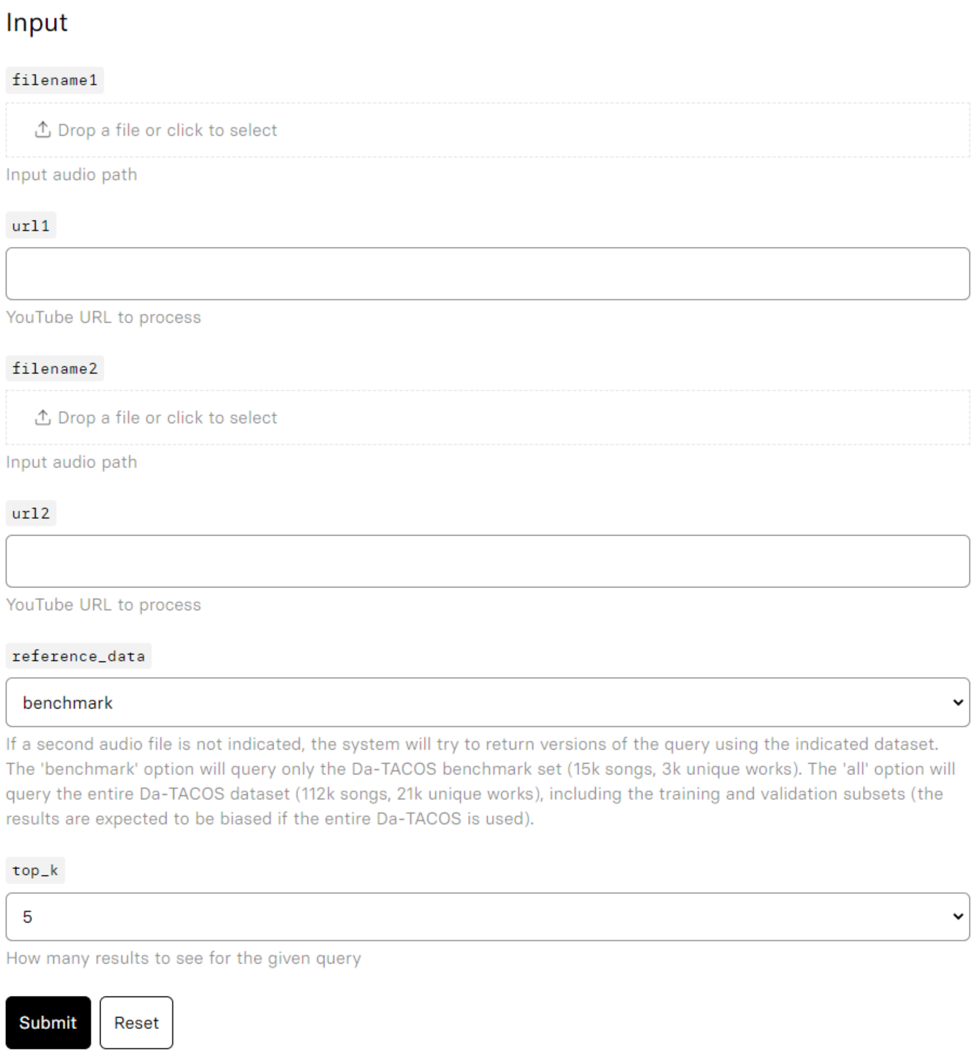
**System demo** (https://replicate.ai/mtg/musical-version-identification) — We created an online demo of one of our VI systems using the templates from Replicate.ai[60] (see Figure A.3). Our system demo provides two use cases:

- Users can provide two tracks by either uploading audio files or by indicating YouTube links. We then extract the embeddings of the both tracks and compute a similarity score. The resulting score is compared to similarity distributions of version and non-version pairs computed using the Da-TACOS benchmark subset (see Figure A.4).
- Users can provide a single track by either uploading an audio file or by indicating a YouTube link. We then extract the embedding of that track and compute its similarity against a reference corpus that contains precomputed embeddings. Users can choose between the Da-TACOS benchmark subset or the entire Da-TACOS dataset as the reference corpus. The results are returned in a table format (see Figure A.5).

**Tutorial** (https://bit.ly/versions-tutorial) — We presented a tutorial on musical version identification in the scope of the International Society of Music Information Retrieval Conference (ISMIR) in 2020[61]. The presentation slides are publicly available.

[60] https://replicate.ai/

[61] https://program.ismir2020.net/tutorials.html


Input

filename1

Drop a file or click to select

Input audio path

url1

YouTube URL to process

filename2

Drop a file or click to select

Input audio path

url2

YouTube URL to process

reference_data

benchmark

If a second audio file is not indicated, the system will try to return versions of the query using the indicated dataset. The 'benchmark' option will query only the Da-TACOS benchmark set (15k songs, 3k unique works). The 'all' option will query the entire Da-TACOS dataset (112k songs, 21k unique works), including the training and validation subsets (the results are expected to be biased if the entire Da-TACOS is used).

top_k

5

How many results to see for the given query

**Figure A.3:** The input options for the system demo based on the template provided by Replicate.ai.

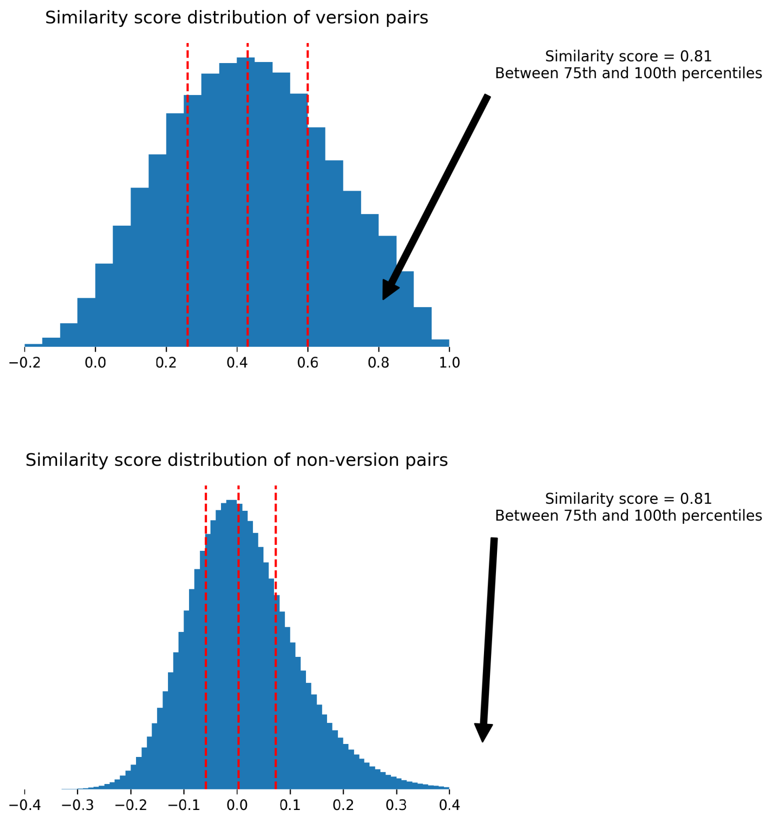
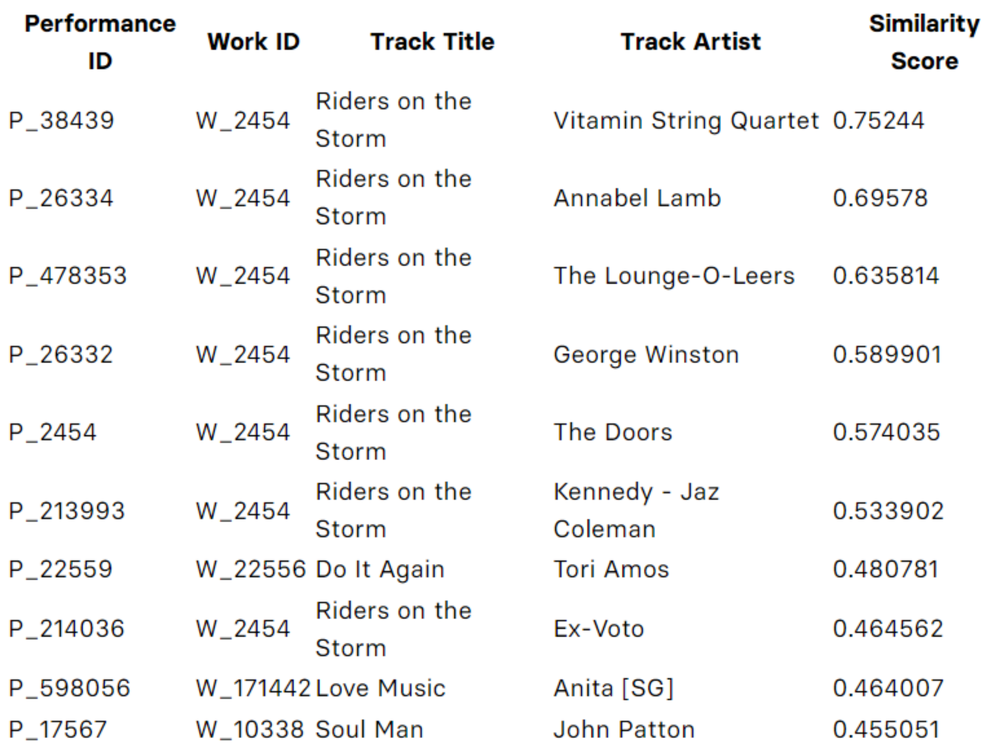


**Figure A.4:** The output of the system demo obtained by comparing two versions of the track "Like a Rolling Stone" by Bob Dylan and John Mayer.



| Performance ID | Work ID | Track Title | Track Artist | Similarity Score |
|---|---|---|---|---|
| P_38439 | W_2454 | Riders on the Storm | Vitamin String Quartet | 0.75244 |
| P_26334 | W_2454 | Riders on the Storm | Annabel Lamb | 0.69578 |
| P_478353 | W_2454 | Riders on the Storm | The Lounge-O-Leers | 0.635814 |
| P_26332 | W_2454 | Riders on the Storm | George Winston | 0.589901 |
| P_2454 | W_2454 | Riders on the Storm | The Doors | 0.574035 |
| P_213993 | W_2454 | Riders on the Storm | Kennedy - Jaz Coleman | 0.533902 |
| P_22559 | W_22556 | Do It Again | Tori Amos | 0.480781 |
| P_214036 | W_2454 | Riders on the Storm | Ex-Voto | 0.464562 |
| P_598056 | W_171442 | Love Music | Anita [SG] | 0.464007 |
| P_17567 | W_10338 | Soul Man | John Patton | 0.455051 |

**Figure A.5:** The output of the system demo obtained by retrieving top-10 results from the benchmark subset of Da-TACOS using a version of the track "Riders on the Storm" performed by Dezperadoz.