Learning Audio Representations for Cross-Version Retrieval of Western Classical Music

Lernen von Audiodarstellungen für die versionsübergreifende Suche westlicher klassischer Musik

Dissertation

Der Technischen Fakultät

der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur

Erlangung des Doktorgrades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von

Frank Zalkow

aus

Großenhain

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	28. Juni 2021
Vorsitzender des Promotionsorgans:	Prof. Dr.-Ing. Knut Graichen
1. Gutachter:	Prof. Dr. rer. nat. Meinard Müller
2. Gutachter:	Prof. Dr.-Ing. Sebastian Stober

Abstract

Ongoing digitization efforts lead to vast amounts of music data, e.g., audio and video recordings, symbolically encoded scores, or graphical sheet music. Accessing this data in a convenient way requires flexible retrieval strategies. One access paradigm is known as "query by example," where a short music excerpt in a specific representation is given as a query. The task is to automatically retrieve documents from a music database that are similar to the query in certain parts or aspects. This thesis addresses two different cross-version retrieval scenarios of Western classical music, where the aim is to find the database's audio recordings that are based on the same musical work as the query. Depending on the respective scenario, one requires task-specific audio representations to compare the query and the database documents. Various approaches for learning such audio representations with deep neural networks are proposed, leading to improvements in the efficiency of the search and the quality of the retrieval results.

In the first scenario, the query is a short audio snippet. The retrieval is based on audio shingles, which are short sequences of chroma features capturing properties of the harmonic and melodic content of the audio recordings. The comparison between the query and the recordings from the database is realized by a nearest-neighbor search of the audio shingles. The thesis contains various contributions to increase the efficiency of the retrieval procedure in this scenario. In order to reduce the dimensionality of the shingles, deep-learning-based embedding techniques are used. Furthermore, a graph-based index structure for efficient nearest-neighbor search is applied. These adaptations lead to substantial improvements in terms of runtime and memory requirements.

In the second scenario, a symbolically encoded monophonic musical theme is used as a query. The retrieval is based on a sequence-alignment algorithm relying on chroma-based audio features. This scenario is more challenging than the first one because the query (monophonic symbolic theme) and the database documents (audio recordings of polyphonic music) are fundamentally different from each other. The thesis contains various contributions to improve the retrieval results in this scenario. On the one hand, a novel dataset for musical themes is introduced that is helpful for evaluation purposes and supervised training procedures. On the other hand, various enhanced chroma representations are proposed for the retrieval task. In particular, a novel chroma-feature variant is introduced, where theme-like structures in the musical content of the audio recordings are enhanced by a deep neural network trained with a loss function (CTC) that allows for aligning the themes to the audio recordings during the training procedure. The experiments described in this thesis show that the results of the theme-based retrieval task are substantially improved by using this representation.

Zusammenfassung

Fortwährende Digitalisierungsbemühungen führen zu großen Mengen an Musikdaten, z. B. Audio- oder Videoaufnahmen sowie symbolisch oder grafisch kodierte Notentexte. Um bequem auf diese Daten zuzugreifen, werden flexible Suchverfahren benötigt. Ein Zugriffsparadigma ist die „Anfrage anhand von Beispielen“, bei dem ein kurzer Musikausschnitt in einer spezifischen Darstellung als Anfrage verwendet wird. Die Aufgabe besteht darin, automatisch Dokumente in einer Musikdatenbank zu finden, die der Anfrage ähneln. Diese Dissertation befasst sich mit zwei versionsübergreifenden Suchszenarien für westliche klassische Musik, bei denen es darum geht, Audioaufnahmen in einer Datenbank zu finden, die auf demselben Musikwerk wie die Anfrage basieren. Abhängig vom jeweiligen Szenario sind aufgabenspezifische Audiodarstellungen erforderlich, um die Anfrage und die Datenbankdokumente zu vergleichen. Es werden verschiedene Ansätze zum Lernen solcher Darstellungen mit neuronalen Netzwerken vorgestellt, welche die Effizienz der Suche und die Qualität der Ergebnisse verbessern.

Im ersten Szenario ist die Anfrage ein kurzer Audioausschnitt. Die Suche basiert auf Audio-Schindeln – kurze Sequenzen von Chroma-Merkmalen, welche harmonische und melodische Eigenschaften des Audioinhalts erfassen. Um die Anfrage mit den Aufnahmen aus der Datenbank zu vergleichen, wird eine Nachbarschaftssuche der Audio-Schindeln durchgeführt. In der Dissertation werden verschiedene Beiträge zur Steigerung der Effizienz des Suchverfahrens in diesem Szenario vorgestellt. Um die Dimensionalität der Schindeln zu verringern, werden auf neuronalen Netzwerken basierende Einbettungstechniken angewendet. Außerdem wird eine graphenbasierte Indexstruktur für die effiziente Nachbarschaftssuche eingesetzt. Diese Anpassungen führen zu erheblichen Verbesserungen hinsichtlich Laufzeit und Speicherbedarf.

Im zweiten Szenario wird ein symbolisch codiertes monophones Thema als Anfrage verwendet, wobei die Suche auf der Sequenzalignierung von Chroma-Merkmalen basiert. Das Szenario ist anspruchsvoller als das erste, da sich die Anfrage (monophones symbolisches Thema) und die Datenbankdokumente (Audioaufnahmen polyphoner Musik) grundlegend unterscheiden. In der Dissertation werden verschiedene Beiträge zur Verbesserung der Suchergebnisse in diesem Szenario vorgestellt. Zum einen wird ein neuer Datensatz für Themen vorgestellt, der Auswertungszwecken und überwachten Trainingsprozeduren dient. Zum anderen werden verschiedene verbesserte Chroma-Merkmale für die Aufgabe vorgeschlagen. Insbesondere wird eine neue Chroma-Variante eingeführt, bei der themenähnliche Strukturen in den Audiodaten durch ein neuronales Netzwerk verstärkt werden. Dieses wird mit einer Verlustfunktion (CTC) trainiert, die eine Alignierung von Themen und Aufnahmen während des Trainings ermöglicht. Experimente zeigen, dass die Ergebnisse der themenbasierten Suchaufgabe durch Verwendung dieser Darstellung wesentlich verbessert werden.

Contents

Abstract i

Zusammenfassung iii

1 Introduction 1
- 1.1 Structure and Main Contributions of this Thesis . . . 3
- 1.2 Main Publications . . . 4
- 1.3 Additional Publications . . . 5
- 1.4 Acknowledgments . . . 6

2 Fundamentals of Audio Representations 9
- 2.1 Waveform . . . 9
- 2.2 Fourier Transform and Short-Time Fourier Transform . . . 11
- 2.3 Log-Frequency Spectrogram and Chromagram . . . 14
- 2.4 IIR-Filter-Based Time-Frequency Transform . . . 16
- 2.5 Constant-Q Transform and Harmonic Constant-Q Transform . . . 18

Part I Learning Low-Dimensional Shingle Embeddings 23

3 Learning Low-Dimensional Embeddings of Audio Shingles for Cross-Version Music Retrieval 25
- 3.1 Introduction . . . 25
- 3.2 Related Work . . . 27
- 3.3 Shingle-Based Retrieval Scenario . . . 29
- 3.4 Shingle Embedding . . . 31
  - 3.4.1 Brute Force . . . 31
  - 3.4.2 PCA . . . 32
  - 3.4.3 Neural Network with Triplet Loss . . . 32
- 3.5 Basic Experiments . . . 34
  - 3.5.1 Training and Testing Datasets . . . 34
  - 3.5.2 Evaluation Procedure . . . 36
  - 3.5.3 Brute Force . . . 36

3.5.4 PCA . . . 38
3.5.5 Neural Network with Triplet Loss . . . 39
3.5.6 Influence of α . . . 39
3.5.7 Runtime Experiment . . . 40
3.6 Extended Experiments . . . 42
3.6.1 Extended Dataset . . . 43
3.6.2 Evaluation . . . 44
3.6.3 Dependency on Query Length . . . 45
3.6.4 Distance Analysis . . . 47
3.7 Conclusions . . . 50

4 Efficient Cross-Version Music Retrieval Using Graph-Based Index Structures 53
4.1 Introduction . . . 54
4.2 Music Retrieval Application . . . 56
4.2.1 Motivating Retrieval Scenario . . . 56
4.2.2 Cross-Version Retrieval System . . . 57
4.2.3 Datasets . . . 58
4.3 Graph-Based Nearest Neighbor Search . . . 59
4.3.1 Graph-Based Data Structures . . . 59
4.3.2 HNSW Graphs . . . 60
4.4 Experiments . . . 62
4.4.1 Experimental Setup . . . 62
4.4.2 Retrieval Quality . . . 64
4.4.3 Feature Computation . . . 65
4.4.4 Constructing, Saving, and Loading the Index . . . 66
4.4.5 Retrieval Time . . . 68
4.5 Conclusions . . . 70

Part II Learning Theme-Based Salience Representations 73

5 MTD: A Multimodal Dataset of Musical Themes for MIR Research 75
5.1 Introduction . . . 75
5.2 Related Datasets and Literature . . . 77
5.2.1 Datasets . . . 77
5.2.2 Literature . . . 79
5.3 Musical Themes in Western Classical Music . . . 80
5.4 Dataset . . . 81
5.4.1 BM and EDM . . . 82
5.4.2 Metadata . . . 82
5.4.3 Symbolic Encodings . . . 85

5.4.4 Audio Recordings . . . 86
5.4.5 Alignment Data . . . 87
5.4.6 Directory and File Structure . . . 87
5.5 Interfaces and Tools . . . 88
5.6 Applications and Future Work . . . 90

6 Evaluating Salience Representations for Cross-Modal Music Retrieval 93
6.1 Introduction . . . 93
6.2 Dataset . . . 95
6.3 Feature Representations . . . 95
6.4 Experiments . . . 98
6.4.1 Retrieval Procedure . . . 98
6.4.2 Retrieval Results . . . 99
6.4.3 Discussion of Matching Quality . . . 100
6.5 Conclusions . . . 101

7 CTC-Based Learning of Deep Chroma Features for Cross-Modal Music Retrieval 103
7.1 Introduction . . . 104
7.2 Cross-Modal Retrieval Application . . . 106
7.2.1 Related Work . . . 106
7.2.2 Dataset . . . 107
7.2.3 Basic Retrieval Procedure . . . 107
7.3 Deep Salience and Deep Chroma Models . . . 108
7.3.1 Related Work . . . 108
7.3.2 Deep Salience Model Adaptation . . . 108
7.4 CTC Loss . . . 110
7.4.1 Loss Computation . . . 110
7.4.2 Chroma Feature Computation . . . 112
7.5 Retrieval Experiments . . . 113
7.5.1 Baseline Experiments . . . 113
7.5.2 Training Details . . . 114
7.5.3 CTC-Based Results . . . 115
7.5.4 Oracle Experiment . . . 115
7.5.5 Representative Example . . . 116
7.6 Effect of CTC Loss . . . 117
7.6.1 Comparison with Linear Scaling . . . 117
7.6.2 Comparison with Strongly Aligned Data . . . 118
7.6.3 Qualitative Comparison . . . 118
7.7 Temporal Granularity . . . 119

7.7.1 Granularity Measure . . . 120
7.7.2 Effect on Retrieval . . . 121
7.8 Musical Evaluation . . . 122
7.8.1 Musical Texture . . . 122
7.8.2 Analysis of Retrieval Results . . . 122
7.8.3 Examples . . . 123
7.9 Future Work and Conclusions . . . 126
7.9.1 Future Work . . . 126
7.9.2 Conclusions . . . 126

8 Summary and Future Work **129**

Appendix **131**
A Frequency Responses Used in the STFT-Based Log-Frequency Spectrogram Computation . . . 131
B Connectionist Temporal Classification . . . 133
B.1 Introduction . . . 133
B.2 CTC Loss Computation . . . 133
B.3 Dynamic Programming . . . 135
B.4 Toy Example . . . 138
B.5 Musical Example . . . 139

Abbreviations **141**

Bibliography **143**

1 Introduction

When talking about music, an essential notion is the "musical work"—a widely used term, which may refer to different things, such as a musical score, a performance, or a recording. However, thinking about it more closely, it may not be obvious what the term means because none of these things is identical to a musical work. According to the widespread consensus in philosophy [103], a musical work can be understood as an abstract object, which manifests in different instances, such as scores, performances, or recordings. To a certain degree, this philosophical perspective is reflected in the multimodal nature of music data. A given musical work is often available in many different modalities, for example, as audio or video recordings, symbolic representations, or graphically encoded sheet music [128]. Given this variety of music data, music constitutes a rich multimedia domain with many exciting applications and tasks [108]. For example, comparing various representations or "versions" of a musical work is a challenging task that requires methods from different research disciplines such as information retrieval, signal processing, and machine learning. These three research areas are relevant for this thesis, which presents various contributions for searching and comparing different versions of the same musical work.

We address two different music information retrieval (MIR) scenarios based on the query-by-example paradigm, where a short music excerpt in a specific representation is given as a query, and we aim to retrieve relevant documents from a database that are similar to this query in certain aspects [40]. In our scenarios, we search for recordings of Western classical music based on the same musical work as the query music excerpt. Depending on the type of query used in the respective task, we need to derive representations of the query and the database recordings that make them comparable. In traditional signal processing, one uses manually designed feature computation procedures to obtain audio representations that are suited for the retrieval task [128]. Using more recent supervised machine-learning approaches, one derives task-specific data representations through computation procedures that are automatically learned by employing annotated training data [17, 225]. In recent years, deep learning has become a powerful machine-learning paradigm for multimedia processing in general [224] and audio processing in particular [148].

Figure 1.1 illustrates the two retrieval tasks considered in this thesis, with the respective proposed approaches for learning task-specific audio representations and retrieving relevant recordings from a database. In our first retrieval scenario, we are given a short audio snippet of Western classical music as a query. The aim is to retrieve all audio recordings from a database that are based on the same musical work as the query. This scenario constitutes a cross-version retrieval task, where the query

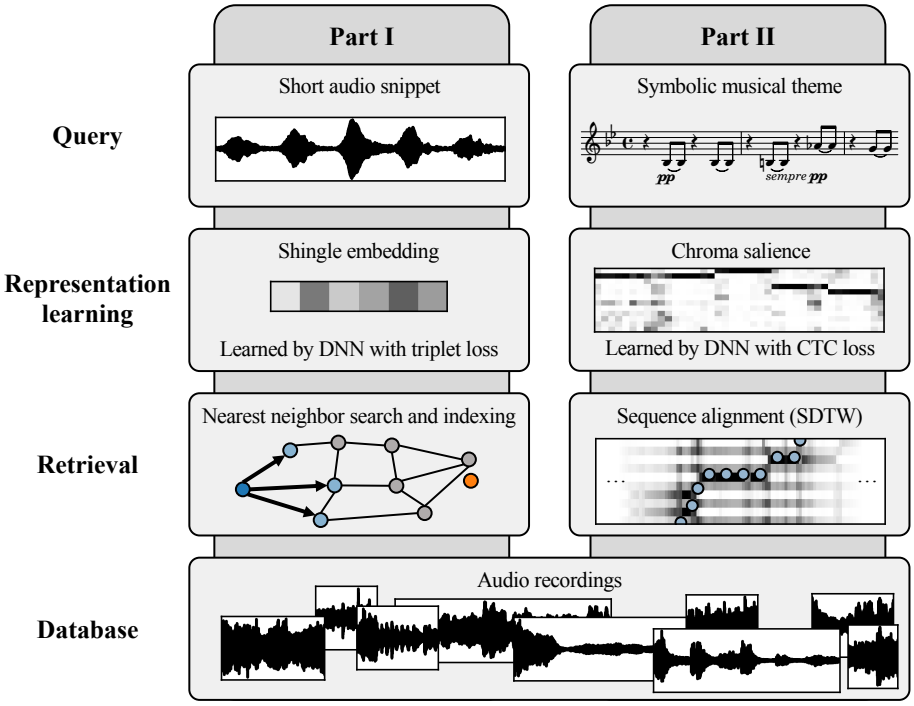


Figure 1.1: Illustration of the main tasks and techniques of Part I and Part II in this thesis.

and the relevant database documents are recordings of different performances (versions) of the same musical work. We build upon previous work, which is based on a nearest neighbor search using short sequences of chroma features (which capture properties of the harmonic and melodic content of the audio recordings), also referred to as audio shingles. In terms of the quality of the retrieval results, this task can be considered close to being solved by the approach used (for certain kinds of music). Besides the quality of the results, the efficiency of the retrieval procedure is also an important aspect. Two main contributions of this thesis are approaches to accelerate the search for this task. First, we adapt two embedding techniques to reduce the dimensionality of the audio shingles, one based on classical principal component analysis (PCA) and another based on deep neural networks (DNNs), trained with a special loss function for representation learning, referred to as triplet loss. The learned shingle embeddings lead to decreased memory and disk space requirements and enable the usage of standard index structures for nearest neighbor search, such as K-d trees. Second, we explore the potential of a modern graph-based index, denoted as hierarchical navigable small world (HNSW) graph to decrease the runtime of the search further. Overall, our contributions show how the retrieval task's efficiency can be increased by several orders of magnitude.

Our second scenario is a cross-version retrieval task, where the different versions of a musical work refer to different kinds of music representations instead of different performances. The query is a symbolically encoded monophonic musical theme, and the task is to find the audio recordings in the database, where the respective theme is being played (typically in a polyphonic context with other musical voices). This scenario poses several challenges because of fundamental differences between the query

and the database documents. For example, these differences relate to the modality (symbolic vs. audio), the degree of polyphony (monophonic vs. polyphonic), and several performance aspects (e.g., tempo and tuning). To make the different modalities comparable, we derive an audio representation of the database documents that is similar to the query representation. In particular, we compute musical features by adapting melody-enhanced pitch salience representations. Furthermore, we propose a task-specific salience representation for musical themes. To this end, we use DNNs that need to implicitly enhance theme-like structures and attenuate all other musical voices in the audio recordings. Standard training procedures for DNNs usually require training pairs with local alignments, e.g., musical themes and audio recordings with annotations for the time positions of the theme's notes. In our contribution, we use pairs of musical themes and audio recordings without local alignments (i.e., without annotations for the time positions of the theme's notes). These weakly aligned training pairs pose a fundamental challenge when used to train DNNs. We solve this problem by implicitly aligning the themes to the audio recordings during the training process. To this end, we employ a particular loss function, referred to as connectionist temporal classification (CTC) loss, to learn audio representations that are suited for the retrieval task. These features are then used in the retrieval procedure based on a sequence-alignment algorithm referred to as subsequence dynamic time warping (SDTW). We analyze the CTC-based features in-depth and show that they improve the state of the art in the theme-based retrieval application.

In summary, this thesis addresses two different cross-version retrieval scenarios based on the query-by-example paradigm. A common aspect of both scenarios is the search for relevant audio recordings of Western classical music in a database. The main difference is the type of query used (short audio snippet or symbolically encoded musical theme), leading to different strategies for learning task-specific audio representations and retrieving audio recordings from the database. The first retrieval scenario is less challenging because the query and the database documents are similar (i.e., being audio recordings of polyphonic music). Rather than improving the retrieval quality, our contributions concern the efficiency and speed of the search. The second scenario is more challenging because of the fundamental differences between query (monophonic symbolic theme) and database documents (audio recordings of polyphonic music). Here, our contributions focus on improving the retrieval results despite these differences.

1.1 Structure and Main Contributions of this Thesis

This thesis consists of two parts (see Figure 1.1), according to the two retrieval scenarios considered. Before the beginning of the first part, we summarize some fundamentals of audio representations used throughout this thesis (Chapter 2).

Part I starts with Chapter 3, where we describe our first retrieval scenario in detail. We present two approaches for dimensionality reduction of audio shingles used for cross-version music retrieval, one based on PCA and another one based on DNNs trained with the triplet loss. Furthermore, we discuss the trade-offs between embedding dimensionality, retrieval quality, and efficiency.

In Chapter 4, we explore the HNSW graph for nearest neighbor search to accelerate the cross-version retrieval application. As a main result, we show that this graph increases the search speed of the retrieval system by several orders of magnitude without a substantial negative impact on the retrieval quality in our application.

Part II begins with Chapter 5, where we introduce a novel dataset, denoted by *musical theme dataset* (MTD), containing several digital representations of more than 2000 musical themes. Beyond graphical sheet music, the dataset provides various symbolic encodings, audio occurrences, alignment annotations, and metadata.

Then, in Chapter 6, we describe our second retrieval scenario, which is based on symbolic musical themes. To compare the themes with the audio recordings, we propose to employ melody-enhanced pitch salience representations. We adapt, compare, and evaluate several conceptually different salience representations from the literature for our cross-modal retrieval scenario.

In Chapter 7, we then learn a task-specific salience representation using the MTD as training data. In particular, we show how to apply the CTC loss in the training procedure, which uses weakly aligned data. This data consists of symbolic themes and audio recordings, where only the time positions of the beginning and end of a theme occurrence are annotated in an audio recording, without the need for local alignment annotations. We analyze and evaluate the resulting features in our theme-based retrieval scenario and show that they substantially improve the results for this task.

Finally, this thesis is concluded in Chapter 8 with a summary and future work directions.

1.2 Main Publications

The main contributions of this thesis are based on the following publications, which either appeared in peer-reviewed conference proceedings or journals in the fields of audio signal processing and music information retrieval (MIR), or have been submitted to journals and are currently under review.

[215] Frank Zalkow and Meinard Müller. Using weakly aligned score–audio pairs to train deep chroma models for cross-modal music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 184–191, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245400.

[216] Frank Zalkow and Meinard Müller. Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music. *Applied Sciences*, 10(1), 2020. doi: 10.3390/app10010019.

[217] Frank Zalkow and Meinard Müller. CTC-based learning of deep chroma features for score–audio music retrieval. 2021. Currently under review.

[221] Frank Zalkow, Stefan Balke, and Meinard Müller. Evaluating salience representations for cross-modal retrieval of Western classical music recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 331–335, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8683609.

[222] Frank Zalkow, Stefan Balke, Vlora Arifi-Müller, and Meinard Müller. MTD: A multimodal dataset of musical themes for MIR research. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1):180–192, 2020. doi: 10.5334/tismir.68.

[223] Frank Zalkow, Julian Brandner, and Meinard Müller. Efficient retrieval of music recordings using graph-based index structures. *Signals*, 2(2):336–352, 2021. doi: 10.3390/signals2020021.

1.3 Additional Publications

The following publications are also related to audio or music processing but are not considered in this thesis.

[70] Prachi Govalkar, Johannes Fischer, Frank Zalkow, and Christian Dittmar. A comparison of recent neural vocoders for speech signal reconstruction. In *Proceedings of the ISCA Speech Synthesis Workshop (SSW)*, pages 7–12, Vienna, Austria, 2019. doi: 10.21437/SSW.2019-2.

[91] Stephanie Klauk and Frank Zalkow. Das italienische Streichquartett im 18. Jahrhundert. Möglichkeiten der semiautomatisierten Stilanalyse. In *Proceedings of the Jahrestagung der Gesellschaft für Musikforschung (GfM)*, Halle/Saale, Germany, 2015.

[92] Stephanie Klauk and Frank Zalkow. Methoden computergestützter melodischer Analyse am Beispiel italienischer Streichquartette. In Stephanie Klauk, editor, *Instrumentalmusik neben Haydn und Mozart. Analyse, Aufführungspraxis und Edition*, pages 151–168. Saarbrücker Studien zur Musikwissenschaft 20, Königshausen & Neumann, 2020.

[97] Michael Krause, Frank Zalkow, Julia Zalkow, Christof Weiß, and Meinard Müller. Classifying leitmotifs in recordings of operas by Richard Wagner. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 473–480, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245472.

[130] Meinard Müller and Frank Zalkow. FMP Notebooks: Educational material for teaching and learning fundamentals of music processing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 573–580, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527872.

[135] Meinard Müller, Helmut Hedwig, Frank Zalkow, and Stefan Popescu. Constraint-based time-scale modification of music recordings for noise beautification. *Applied Sciences*, 8(3), 2018. doi: 10.3390/app8030436.

[172] Hendrik Schreiber, Frank Zalkow, and Meinard Müller. Modeling and estimating local tempo: A case study on Chopin's mazurkas. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 773–779, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245546.

[204] Christof Weiß, Frank Zalkow, Meinard Müller, Stephanie Klauk, and Rainer Kleinertz. Versionsübergreifende Visualisierung harmonischer Verläufe: Eine Fallstudie zu Wagners Ring-Zyklus. In *Proceedings of the Jahrestagung der Gesellschaft für Informatik (GI)*, pages 205–217, Chemnitz, Germany, 2017. doi: 10.18420/in2017_14.

[205] Christof Weiß, Frank Zalkow, Vlora Arifi-Müller, Meinard Müller, Hendrik Vincent Koops, Anja Volk, and Harald G. Grohganz. Schubert Winterreise dataset: A multimodal scenario for music analysis. *ACM Journal on Computing and Cultural Heritage (JOCCH)*, 14(2), 2021. doi: 10.1145/3429743.

[214] Frank Zalkow and Meinard Müller. Vergleich von PCA- und Autoencoder-basierter Dimensionsreduktion von Merkmalssequenzen für die effiziente Musiksuche. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1526–1529, Munich, Germany, 2018.

[218] Frank Zalkow, Stephan Brand, and Bejamin Graf. Musical style modification as an optimization problem. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 206–211, Utrecht, The Netherlands, 2016.

[219] Frank Zalkow, Christof Weiß, and Meinard Müller. Exploring tonal-dramatic relationships in Richard Wagner's Ring cycle. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 642–648, Suzhou, China, 2017. doi: 10.5281/zenodo.1415760.

[220] Frank Zalkow, Christof Weiß, Thomas Prätzlich, Vlora Arifi-Müller, and Meinard Müller. A multi-version approach for transferring measure annotations between music recordings. In *Proceedings of the AES International Conference on Semantic Audio*, pages 148–155, Erlangen, Germany, 2017.

1.4 Acknowledgments

The paths of my supervisor Meinard Müller and me crossed long before I started working on this thesis. Being interested in technology and music, I studied at the Institute for Music Informatics and Musicology at the University of Music in Karlsruhe, Germany. The institute organized a weekly colloquium with exciting speakers from various domains, such as musicology, sonic arts, or computer science. During summer term 2011, while I still was an undergraduate student, the colloquium featured a talk entitled "Beethoven, Bach, and Billions of Bytes" by Meinard Müller. The lecture gave a kind introduction to music processing and MIR. After attending, I was enthusiastic and immediately thought that I would like to get involved in this kind of research. I contacted Meinard and was finally allowed to visit his bi-weekly music processing lecture in Bonn in the summer term of 2012. During this time, Jonathan Driedger, a Ph.D. student from Meinard at the time, regularly offered me overnight stays such that my student budget would allow for the regular visits. Thanks to the warm welcome from Meinard and his team, my trips to Bonn were a great pleasure, not only from a scientific perspective, and gently paved my way in getting involved with MIR. It then took a couple of years for me to finish my Bachelor's and Master's degree. In August of 2016, I started my work as a Ph.D. student in Meinard's group, which was then based at the International Audio Laboratories Erlangen.

I want to express my greatest thanks to Meinard, who introduced me to the research field of MIR, who supported me for so long (long before I joined his group and afterward), and who supervised me in an extremely intensive, friendly, and instructive manner. Sometimes we had hard times together, we often had a lot of fun, and, above all, I had the privilege of learning from and gaining scientific insights with him. Without his enduring support, this thesis would not have been possible. Thank you!

While working in Meinard's group, I had the opportunity to work with great colleagues. My closest cooperators were the members of "GroupMM," whom I want to thank (in alphabetic order): Vlora Arifi-Müller, Stefan Balke, Christian Dittmar, Jonathan Driedger, Michael Krause, Patricio López-Serrano, Peter Meier, Thomas Prätzlich, Sebastian Rosenzweig, Hendrik Schreiber, and Christof Weiß. I also enjoyed working with many students, who I partly supervised and from who I was also learning. Among the many smart students, I am grateful to have worked with, I would like to name (in alphabetic order) Julian Brandner, Lu Cheng, Helmut Hedwig, Lena Krauß, Anna-Luisa Römling, Quirin Seilbeck, and

Angel Villar-Corrales. I am also thankful to have met various researchers cooperating with GroupMM, especially Stephanie Klauk, Rainer Kleinertz, Hendrik Vincent Koops, and Frank Scherbaum.

I also want to thank the great AudioLabs team from other groups who have been helpful in so many ways, ranging from scientific discussions, to lectures on how to make good coffee, and to great help with administrative issues. Many thanks (again in alphabetic order) to Alexander Adami, Carlotta Anemüller, Stefan Bayer, Sebastian Braun, Tom Bäckström, Soumitro Chakrabarty, Alexandra Craciun, Pablo Delgado, Sascha Dick, Bernd Edler, Youssef El Baba, Esther Feichtner, Johannes Fischer, Richard Füg, Ning Guo, Emanuël Habets, Tracy Harris, Jürgen Herre, Adrian Herzog, Thorsten Kastner, Michael Krause, Ankit Kumar, María Luis Valero, Wolfgang Mack, Goran Markovic, Daniele Mirabilii, Nils Peters, Day-See Riechmann, Sebastian Schlecht, Konstantin Schmidt, Martin Strauß, Fabian-Robert Stöter, Armin Taghipour, Maja Taseska, Stefan Turowski, Frank Wefers, Elke Weiland, Nils Werner, and Niklas Winter. Furthermore, I gratefully acknowledge the compute resources and support provided by the Erlangen Regional Computing Center (RRZE).

During my Ph.D. studies, I had the opportunity to do an internship at the Fraunhofer IIS. I would like to thank Christian Dittmar, Johannes Fischer, Christian Geishauser, Prachi Govalkar, Luzian Hahn, and Yigitcan Özer for this great experience and the collaborative work.

My work was financially supported by the German Research Foundation (DfG), to which I am very grateful. Some of the DfG-funded projects were carried out with great project partners: the Carus publisher and the Semantic Music Technology (SMT) group of the Fraunhofer IDMT in Ilmenau. I thank the Carus publisher and its team, especially Irmgard Bauer, Mayira Florschütz, Johannes Graulich, Ester Petri, Iris Pfeiffer, Andreas Weller, and Uwe Wolf. I also thank the SMT group, especially Jakob Abeßer, Hanna Lukashevich, Stylianos Ioannis Mimilakis, and Michael Taenzer.

The research field of MIR is not only great because of exciting research questions and insights but also because of people. I am privileged and thankful to have met several inspiring researchers, especially Andreas Arzt, Rachel Bittner, Sebastian Böck, Helena Cuesta, Matthias Dorfer, Masataka Goto, Emilia Gómez, David Lewis, Cynthia Liem, Antoine Liutkus, Brian McFee, Ryo Nishikimi, Geoffroy Peeters, Polina Proutskova, Justin Salamon, Anja Volk, and Yi-Hsuan Yang. I want to make a special mention of Sebastian Stober, who has regularly given me good advice and agreed to be the second reviewer of this thesis. Furthermore, I want to thank Daniel Stoller for fruitful discussions on the CTC loss.

I also thank my family and all my friends for supporting me, believing in me, and giving me completely different perspectives that often helped to rearrange my thoughts. The biggest thanks go to my wife Julia, without whose support this thesis would not have been possible. She prepared the ground for this work by joining me in moving from Karlsruhe to Erlangen, where I wanted to realize my dream of doing a Ph.D. in MIR. I will never forget this! We are expecting twins very soon. I am looking so much forward to them, and I am grateful they did not come too early because I might not be able to give them the right focus if they were already here before I wrote these lines. However, now I am ready for you!

2 Fundamentals of Audio Representations

In this chapter, we introduce the notions of waveforms, time–frequency representations, and chroma features. In particular, we describe several variants of these audio representations. Rather than introducing all mathematical details, we want to convey the main ideas, assuming some basic signal-processing background of the reader. We start by discussing the waveform (Section 2.1), which is the "raw" audio representation. Our first time–frequency transform discussed is the short-time Fourier fransform (STFT), which is of fundamental importance to signal processing (Section 2.2). Having a linear frequency axis, the STFT does not account for the logarithmic frequency perception by humans. To address this issue, we introduce a log-frequency representation by logarithmically scaling the frequency axis of the STFT (Section 2.3). As alternatives, we cover two time–frequency transforms with logarithmic frequency axis, one based on infinite impulse response filters (Section 2.4) and another one, referred to as constant-Q transform (CQT), based on so-called time–frequency atoms (Section 2.5). We also discuss an extension of the latter transform, referred to as harmonic CQT, where the output is a three-dimensional tensor.

2.1 Waveform

The representations presented in this chapter are based on audio signals. Audio signals can be represented as waveforms, which encode the change of air pressure over time. We denote a discrete audio signal, sampled using the sampling frequency $F_{\mathrm{s}} \in \mathbb{R}_{>0}$, by $x\colon\ \mathbb{Z} \to \mathbb{R}$. A value $x(n) \in \mathbb{R}$ at a specific index $n \in \mathbb{Z}$ is also called a sample. In our mathematical model, a signal is of infinite length, having the discrete-time axis $\mathbb{Z}$. In practice, however, we have signals of finite length. Considering the time axis $\mathbb{Z}$, we assume that all samples outside the signal's finite range are zero. For a detailed treatment of audio signals, we refer to the textbook by Müller [128].

Graphically, we can visualize a signal by plotting time–amplitude points $\big(n, x(n)\big)$ over a finite time range. Figure 2.1a shows such a visualization (stem plot) for a signal sampled at $F_{\mathrm{s}} = 22\,050\,\mathrm{Hz}$ that is composed of two components. The first one is a sinusoid having a frequency of 880 Hz and an amplitude of 0.05. The second component is a sudden impulse-like click at sample $n = 11\,025$. We restrict the displayed time to a small range around this sample. We also limit the displayed amplitude range in the figure to $[-0.1, 0.1]$ to show the details of the sinusoidal oscillations. The impulse's amplitude of 1.0 is outside this range.

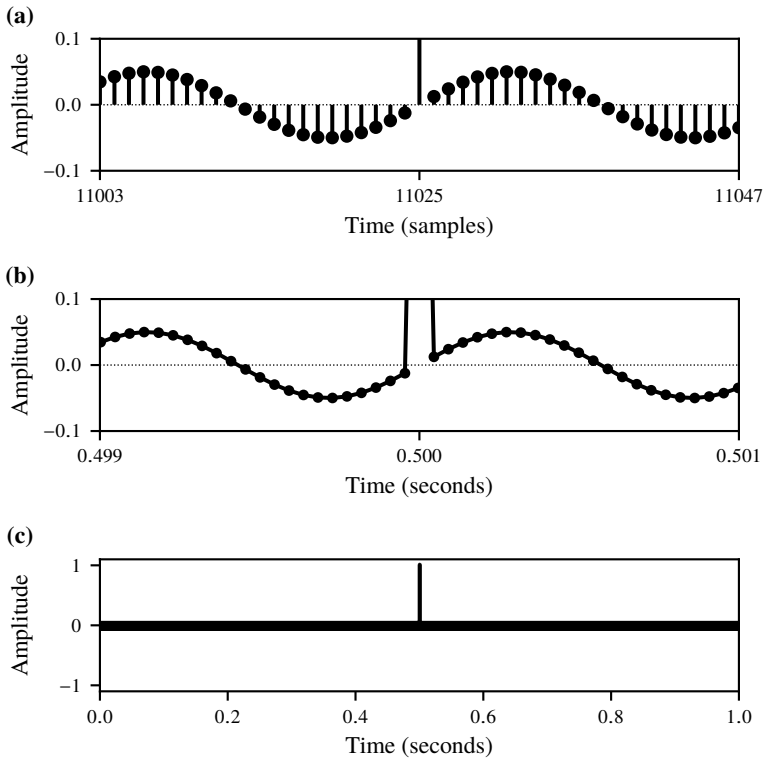


Figure 2.1: Synthetic waveform `Synt` (impulse and sinusoid with amplitude 0.05). **(a)** Stem plot (zoom in). **(b)** Continuous plot (zoom in). **(c)** Continuous plot (zoom out).

As an alternative to the stem plot, we can suggest the impression of a continuous instead of a discrete signal by connecting the plotted points. Furthermore, we use a time axis in seconds and plot the points $\big(n/F_s, x(n)\big)$. We can now better interpret the signal physically. For example, in Figure 2.1b, showing the same signal as before, we now see that the impulse occurs at the second 0.5. Figure 2.1c shows the signal for the time range of 1 second duration and the amplitude range of $[-1, 1]$. Here, the details of the sinusoidal wave are not visible in the visualization because its oscillations are so fast that they appear as a thick black line between amplitude -0.05 and 0.05. On the contrary, the impulse is clearly visible at the second 0.5.

As a real music example, in Figure 2.2a, we show a waveform of an excerpt from a recording of Beethoven's *Great Fugue* Op. 133 by the Guarneri String Quartet. The waveform's complex shape suggests that it is not composed of just a few simple components, as our synthetic example. The sheet music corresponding to this excerpt is shown in Figure 2.2b. In our context, Beethoven's unusual notation (tied eighth notes instead of quarter notes) is not important.[1] When comparing the waveform to the sheet music, we see that the five notes correspond to the five attack–decay phases in the waveform. The amplitude decays close to zero in between each of these phases, which corresponds to the rests in the sheet music. Both examples of Figure 2.1 and Figure 2.2 will re-appear throughout this chapter to illustrate various audio representations.

An important observation is that a waveform is not suited to reveal details about the signal's frequency content. Both examples illustrate this issue since we can neither explicitly see the sinusoid's frequency

[1] This particular notation is a topic of debate among musicologists [104].

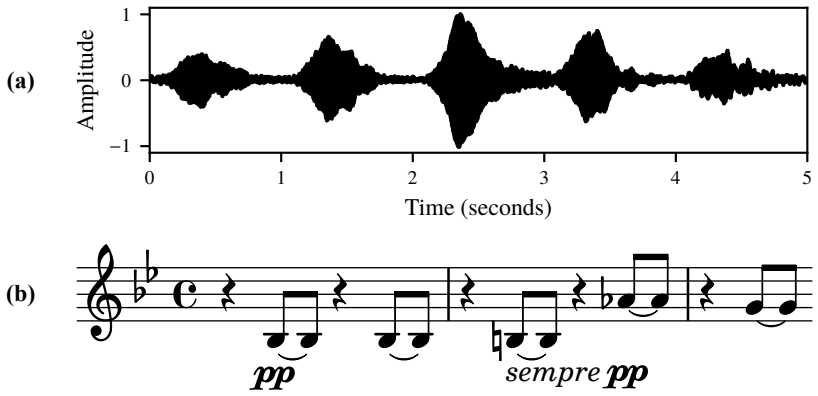


Figure 2.2: Representations for `Beet` (an excerpt from Beethoven's *Great Fugue* Op. 133, beginning of second *Allegro* section, bars 26 et seqq.). **(a)** Waveform of a recording. **(b)** Sheet music.

in the first example, nor do we have insights into the notes' pitches in the second example. This issue motivates us to consider various time–frequency representations in the following.

2.2 Fourier Transform and Short-Time Fourier Transform

Among the most important signal-processing tools are the discrete Fourier transform (DFT) and its local variant, the short-time Fourier fransform (STFT). We describe the main ideas of the DFT and STFT closely following the textbook by Müller [128].

To compute the DFT, we consider a finite temporal range $[0 : N-1] := \{0, \ldots, N-1\}$ of length $N \in \mathbb{N}$ from an audio signal. We compare the signal with N complex exponentials of different frequencies. Loosely speaking, each exponential analyzes the signal with respect to a specific frequency and reveals the components corresponding to that frequency contained in the signal. Mathematically, the DFT is given by

$$\mathcal{X}^{\mathrm{DFT}}(k) = \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n / N) \tag{2.1}$$

for the frequency index $k \in [0 : N-1]$. In our context, only the first $\lfloor N/2 \rfloor + 1$ frequency indices[2] are important because they correspond to the relevant frequency range from 0 Hz (no oscillations, also referred to as DC offset[3]) to the Nyquist frequency (the highest frequency that can be represented with a discrete signal). Intuitively, the magnitude $|\mathcal{X}^{\mathrm{DFT}}(k)|$ of each complex coefficient encodes the extent to which a sinusoid of frequency $k \cdot F_{\mathrm{s}}/N$ Hz is contained in the signal.

Figure 2.3a visualizes the magnitude DFT of our synthetic example (sinusoid and impulse), only showing the lower part of the frequency spectrum. We clearly see a peak at 880 Hz corresponding to the frequency of the sinusoid, having an amplitude of 0.05. The impulse is not visible well (having small frequency components spread over the entire spectrum). Figure 2.3b shows the magnitude DFT for our Beethoven

[2] $\lfloor \cdot \rfloor$ denotes the floor operation (rounding to the nearest integer towards negative infinity) and $\lceil \cdot \rceil$ denotes the ceil operation (rounding to the nearest integer towards positive infinity).

[3] This term originates from electronics, where "DC" refers to a direct current voltage.

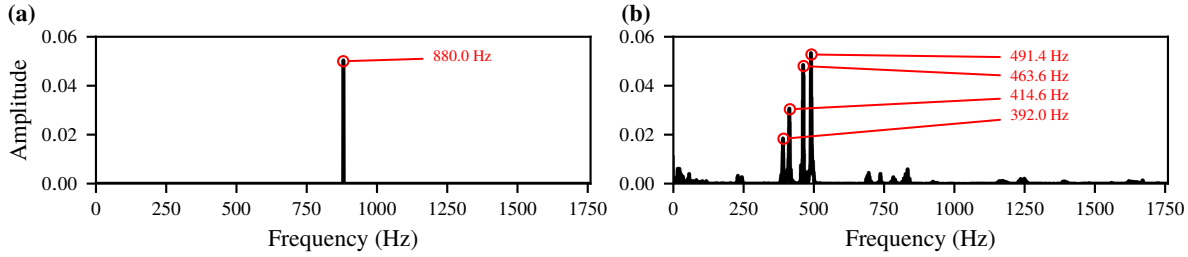


Figure 2.3: Visualizations of the magnitude DFT of **(a)** `Synt`, and **(b)** `Beet`.

example. We see four major peaks in the spectrum at the frequencies of 392.0 Hz, 414.6 Hz, 463.6 Hz, and 491.4 Hz. These peaks result from the notes played, which have a harmonic frequency structure with a fundamental frequency (corresponding to the perceived pitch of the note) and overtones with frequencies that are integer multiples of the fundamental frequency. In the sheet music of our example (Figure 2.2b), we find the notes of $\mathrm{B}_3^\flat$, B_3, $\mathrm{A}_4^\flat$, and G_4.[4] The lower peaks at 392.0 Hz and 414.6 Hz correspond to the fundamental frequency of the notes G_4 and $\mathrm{A}_4^\flat$, respectively. The higher peaks at 463.6 Hz and 491.4 Hz correspond to the first overtones of the notes $\mathrm{B}_3^\flat$ and B_3, respectively. Depending on the note in our example, the fundamental frequency or the first overtone may have the highest amplitude, leading to spectral peaks. In general, our examples show that the DFT captures the frequency content of the signals well but does not encode temporal information explicitly (e.g., *when* a particular note is being played). This shortcoming motivates the usage of time–frequency transforms, such as the STFT.

To compute the STFT, we need to fix a so-called window function $w\colon [0:N-1] \to \mathbb{R}$ of length $N \in \mathbb{N}$, which is used to expose local sections of the signal by suitably shifting the window function in time. These shifts are carried out according to a parameter called hop size $H \in \mathbb{N}$. For a fixed shift, the exposed local section of the signal is processed by the Fourier transform, i.e., it is compared with $\lfloor N/2 \rfloor + 1$ complex exponential functions of different frequencies. The STFT of a signal x is given by

$$\mathcal{X}^{\mathrm{STFT}}(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N) \tag{2.2}$$

with $m \in \mathbb{Z}$ denoting a frame index and $k \in [0:\lfloor N/2 \rfloor]$ a frequency index. Intuitively, the absolute value of the complex Fourier coefficient $\mathcal{X}^{\mathrm{STFT}}(m,k)$ indicates how intensely the frequency of $k \cdot F_\mathrm{s}/N$ Hz is contained in the signal around the time point of $m \cdot H/F_\mathrm{s}$ seconds. This linear correspondence between the parameters m and k with their respective physical interpretations results in a linearly spaced time and frequency grid.

According to Eq. 2.2, the window at $m = 0$ covers a temporal range of $n \in [0:N-1]$. In practice, we take a centric view, where the window at $m = 0$ is centered around $n = 0$. This centric view can be implemented by padding $\lfloor N/2 \rfloor$ samples (having a value of zero) at the beginning of the signal before applying Eq. 2.2.

[4] Here and throughout this thesis, we use the scientific pitch notation, where C_4 refers to middle C.

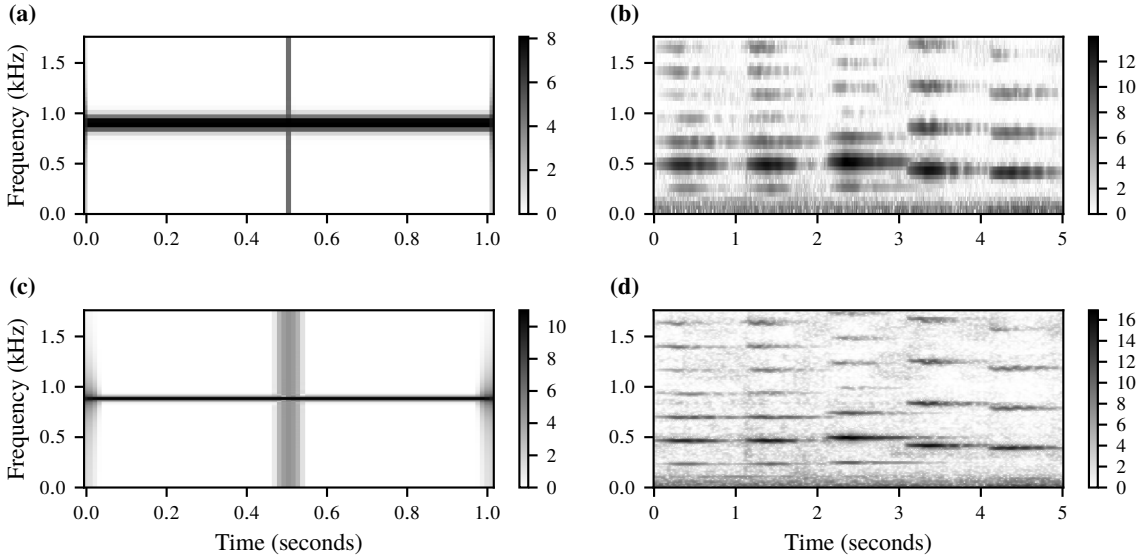


Figure 2.4: Spectrogram visualizations. **(a)** `Synt` ($N = 512$, $H = 256$). **(b)** `Beet` ($N = 512$, $H = 256$). **(c)** `Synt` ($N = 2048$, $H = 256$). **(d)** `Beet` ($N = 2048$, $H = 256$).

We use such a centric view in all windowed transforms described in this chapter. Furthermore, in practice, we use an efficient algorithm to compute the DFT, known as the fast Fourier transform (FFT).

The term "spectrogram" (or "power spectrogram") is often used to denote the squared magnitude of the STFT

$$\mathcal{Y}^{\mathrm{STFT}}(m, k) = |\mathcal{X}^{\mathrm{STFT}}(m, k)|^2. \tag{2.3}$$

In the following, we use the spectrogram to visualize the STFT and to derive musical features. In the visualizations throughout this chapter, we use logarithmic compression in the amplitude domain to enhance small signal components. To this end, we apply the function

$$\Gamma_\gamma(v) = \log(1 + \gamma \cdot v) \tag{2.4}$$

to each value of the spectrogram, using a suitable compression factor $\gamma \in \mathbb{R}_{>0}$. This function yields a positive value $\Gamma_\gamma(v)$ for any positive value $v \in \mathbb{R}_{>0}$.

Figure 2.4a and b visualize spectrograms for our synthetic example and the Beethoven example, respectively. Here, we use a Hann window of about 23.2 ms length and a hop size of about 11.6 ms (512 and 256 samples in a sampling rate $F_{\mathrm{s}} = 22\,050$ Hz, respectively). In the spectrogram of the synthetic example (Figure 2.4a), we can observe the sinusoidal component with a horizontal line and the impulse component with a vertical line. For the Beethoven example (Figure 2.4b), each note is represented by a set of horizontal lines. The lines are roughly equidistant in frequency, corresponding to harmonic structures. We also see that the first three notes have a fundamental frequency with a lower amplitude than the respective first overtone. For the last two notes, however, the fundamental frequency has the highest amplitude. These findings agree with our discussion above about the comparison of the sheet music to the spectral peaks in the magnitude DFT.

The frequency resolution of the STFT is determined by the window size N and the sampling rate F_s (having a spacing of F_s/N Hz). To improve the frequency resolution, we may increase the window size. Figures 2.4c and d show similar visualizations as before, but now with an increased window size of 92.9 ms (2048 samples in a sampling rate $F_s = 22\,050$ Hz). The increased frequency resolution can be seen in the spectrogram of the synthetic example (Figure 2.4c), where the line corresponding to the sinusoid is sharper compared to Figure 2.4a. At the same time, the temporal precision is decreasing, which is reflected by the temporally smoothed line corresponding to the impulse in Figure 2.4c. The increased frequency resolution is evident in the spectrogram of our Beethoven example (Figure 2.4b compared to Figure 2.4d), where the harmonic structures appear much sharper.

2.3 Log-Frequency Spectrogram and Chromagram

The linear frequency axis of the STFT does not account well for the logarithmic frequency perception by humans. To compensate for this shortcoming, we may use the STFT as a basis to derive a representation having a logarithmic frequency axis. To this end, we rescale the STFT's frequency axis according to the logarithmically spaced frequency values of the equal temperament. Such a rescaling can be accomplished in several ways. For example, as we do in the following, one can apply cubic interpolation along the frequency axis of the STFT. The unit of the resulting log-frequency axis (which we also refer to as pitch axis) is given in MIDI note numbers, where the center frequency of a given note number $p \in \mathbb{R}$ is

$$f(p) = f_{\text{ref}} \cdot 2^{(p-69)/12}, \tag{2.5}$$

with $f_{\text{ref}} \in \mathbb{R}$ being the reference frequency for A_4 (usually 440 Hz), associated with the reference pitch $p = 69$. Inspired by the chromatic scale, we usually use a resolution of $B = 12$ bins per octave for our new pitch axis, leading to a resolution of one bin per semitone. A higher $B \in \mathbb{R}$ results in an increased pitch resolution. To have a musically meaningful partition (according to the chromatic scale), B should be a multiple of twelve. In our our case, we use a log-frequency spectrogram with $K \in \mathbb{N}$ pitch bins, where a bin index $k \in [0 : K - 1]$ corresponds to the pitch number

$$p_k = 12 \cdot k/B + p_{\text{min}}, \tag{2.6}$$

with $p_{\text{min}} = 24$ being the lowest pitch considered.

Figures 2.5a and b show log-frequency spectrograms of our running examples, where we use an STFT with a window size of 92.9 ms and a hop size of 11.6 ms ($N = 2048$, $H = 256$, $F_s = 22\,050$ Hz). We use $B = 12$ bins per octave and consider a pitch range of $p \in [24 : 108]$ (C_1 to C_8), resulting in $K = 85$ pitch bins. Using the log-frequency axis, we can verify that the frequency of the sinusoidal component (880 Hz) in our synthetic example (Figure 2.5a) corresponds to the pitch number $p = 81$. In our second example, (Figure 2.5b), one can identify the notes, which correspond to the pitch numbers (58, 58, 59, 68, 67).

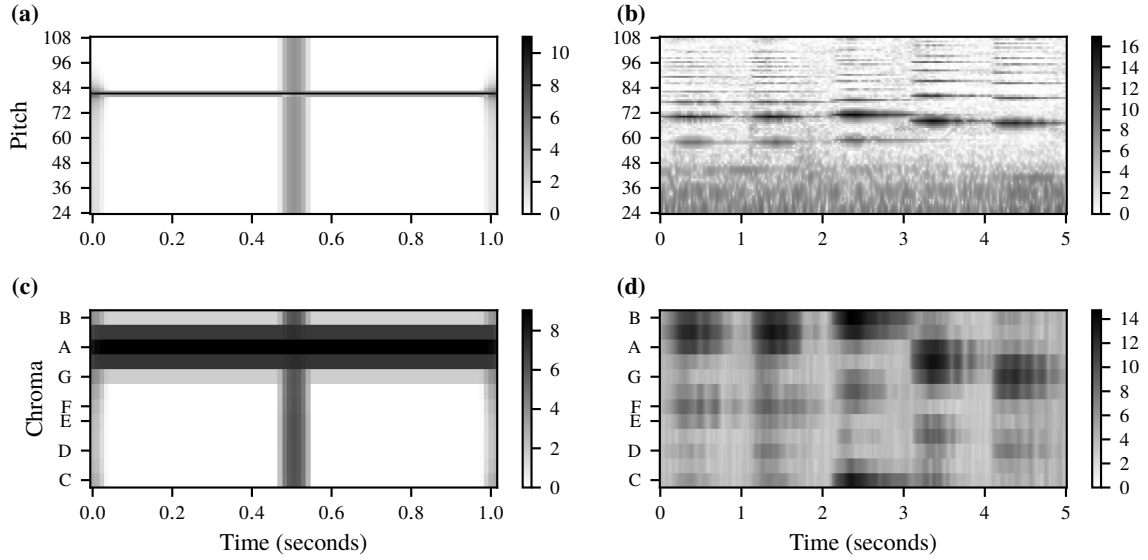


Figure 2.5: STFT-based visualizations ($N = 2048$, $H = 256$). **(a)** Log-frequency spectrogram for `Synt`. **(b)** Log-frequency spectrogram for `Beet`. **(c)** Chromagram for `Synt`. **(d)** Chromagram for `Beet`.

In the next sections, we introduce alternative approaches for computing log-frequency spectrograms, where we also discuss the frequency responses of digital filters that are used in the respective transforms. We can also derive similar frequency responses for the STFT-based log-frequency spectrogram, which we discuss in Appendix A.

In music, we perceive pitches that are one or more octaves apart as musically similar (so-called octave equivalence). Motivated by this musical fact, one often uses in music processing an audio representation called chroma features [14, 68, 128]. This feature type captures the energies contained in the twelve pitch classes $\{\mathrm{C}, \mathrm{C}^\sharp, \mathrm{D}, \ldots, \mathrm{B}\}$ and is robust against octave shifts and (to a certain degree) changes in timbre. There are many ways to compute chroma features. One option is to use the STFT-based log-frequency spectrogram by aggregating its log-frequency grid to the twelve chroma bins. For this aggregation, we add up the amplitude values of all pitch bins that correspond to the same chroma in each frame. Starting with the STFT, we reduce the frequency axis from $\lfloor N/2 \rfloor + 1$ frequency bins to only 12 chroma bins. For more details regarding the chroma computation and their musical background, we refer to the textbook by Müller [128].

Chroma features for our examples are visualized in Figures 2.5c and d. In the chroma representation of the synthetic example (Figure 2.5c), the energy is centered in the A band due to the sinusoid's frequency (880 Hz), corresponding to the note A_5. We also see some energy in the neighboring bins because of a well-known effect called spectral leakage that is related to the windowing of the STFT [128]. The impulse is visible across all chroma bands because it affects all frequency bands. The chroma representation of the Beethoven example (Figure 2.5d) is even more smeared due to the fact that low-pitched components are poorly represented by the STFT with its linear frequency axis. The Beethoven example's pitches ($\mathrm{B}_3^\flat$ to $\mathrm{A}_4^\flat$) are less precisely represented than the sinusoid (A_5) because they are lower. Nevertheless, looking at

the energy center of the note events, we can roughly follow the pitch classes sequence ($B^\flat$, $B^\flat$, B, $A^\flat$, G) in the chromagram.

2.4 IIR-Filter-Based Time-Frequency Transform

As an alternative to the STFT-based log-frequency spectrogram, we may use a transform that yields an inherent logarithmic frequency grid. One such transform is based on filters with infinite impulse response (IIR) and was introduced by Müller [127]. A general introduction to digital filters is beyond the scope of this section, and we only summarize the IIR-based transform assuming some basic signal-processing knowledge of the reader.

In signal processing, a digital filter is a function that enhances or reduces certain aspects of a discrete-time signal. In particular, we consider filters that amplify or attenuate a signal's amplitude in specific frequency bands. We can characterize a filter by its impulse response (the filter output, given an impulse as input) and its frequency response (the Fourier transform of the impulse response). Two main categories are filters with a finite impulse response (FIR) and filters with an infinite impulse response (IIR). In this section, we consider a time–frequency transform that is based on IIR filters. There is a recursive way to implement an IIR filter, described by the difference equation

$$y(n) = -\sum_{\ell_1=1}^{L_1} a(\ell_1)y(n-\ell_1) + \sum_{\ell_2=0}^{L_2} b(\ell_2)x(n-\ell_2), \tag{2.7}$$

where $\big(b(0), b(1), \ldots, b(L_2)\big)$ are referred to as forward filter coefficients (processing the input signal x) and $\big(a(1), \ldots, a(L_1)\big)$ as feedback filter coefficients (processing the output signal y). One may introduce the coefficient $a(0) = 1$, corresponding to the output sample $y(n)$. For more details, we refer to a textbook on signal-processing fundamentals, such as [146].

Beyond the distinction between filters having a finite or infinite impulse response, one may classify filters according to the shape of the filter's frequency response. One such filter category is the bandpass filter, which passes frequencies within a specified range (called passband) around a particular center frequency and attenuates all frequency components outside that range (stopband). The width of the passband is referred to as bandwidth. The bandwidth can be characterized by the Q factor (or quality factor) $Q \in \mathbb{R}$, which denotes the ratio of the passband's center frequency to the bandwidth. Figure 2.6a shows the frequency response of a bandpass filter with a center frequency of 440 Hz and a Q factor of $Q = 25$ (resulting in a bandwidth of $440/25 = 17.6$ Hz). In between the passband and stopband, the response gradually transitions from passing to attenuation (transition band). The small local maxima in the stopbands are referred to as side lobes.

In the IIR-based transform, we use a set of filters (also denoted as filter bank), where each filter is a bandpass filter analyzing the signal components corresponding to a particular pitch, denoted by the

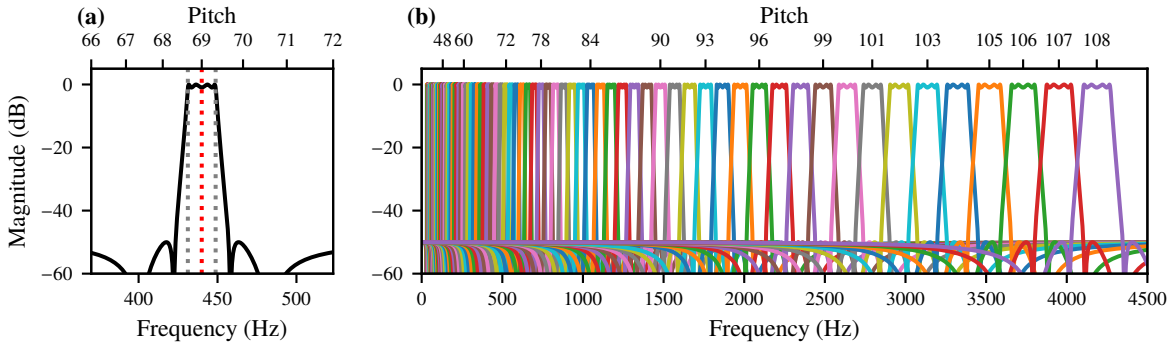


Figure 2.6: **(a)** Frequency response of an IIR bandpass filter with a center frequency of 440 Hz (indicated with dotted red line) and a Q factor of 25 (resulting bandwidth of 17.6 Hz indicated with dotted gray lines). **(b)** Frequency responses for the filterbank used in the IIR-based transform.

pitch number $p \in \mathbb{R}$ with the center frequency $f(p)$ (see Eq. 2.5). As before, we use a pitch range of $p \in [24 : 108]$. In order to analyze the frequency range corresponding to a semitone, the bandwidth of a bandpass filter needs to increase with increasing frequency. To this end, we use a constant Q factor of $Q = 25$ for all filters in our filterbank (when considering $B = 12$ bins per octave). Figure 2.6b shows the frequency responses of all filters used in the IIR-based transform. Due to the logarithmic spacing of the center frequencies, the frequency responses for the lower frequency range visually overlap and can hardly be distinguished. In the upper-frequency range, we can better see the neighboring frequency responses of the bandpass filters.

We denote the output signal of a filter from the filterbank by y_k (using the index $k \in [0 : K - 1]$), corresponding to the pitch p_k and the center frequency $f(p_k)$ (see Eq. 2.5). To capture the energy and decrease the feature rate of a given pitch band y_k, we compute its short-time power, using a sliding window $w : [0 : N - 1] \to \mathbb{R}$ of size $N \in \mathbb{N}$ and a hop size $H \in \mathbb{N}$, similar to the STFT.

$$\mathcal{X}^{\mathrm{IIR}}(m, k) = \sum_{n=0}^{N-1} w(n)|y_k(n + mH)|^2. \tag{2.8}$$

We described the main idea of computing $\mathcal{X}^{\mathrm{IIR}}$ in a simplified form. In practice, we use an implementation, where a multi-resolution procedure is used due to efficiency and numerical stability. In this procedure, different frequency bands are processed with different sampling rates by suitably downsampling the signal before filtering. The different rates are compensated later in the short-time power computation (Eq. 2.8) by using different window and hop sizes. Furthermore, the IIR filters introduce non-linear phase distortions, which are compensated by applying forward–backward filtering. We refer to [127] for more details. In particular, we refer to Figure 3.1 (p. 54) in [127], where the frequency responses are visualized for the different sampling rates, unlike our Figure 2.6b, where we visualized all frequency responses in a single figure.

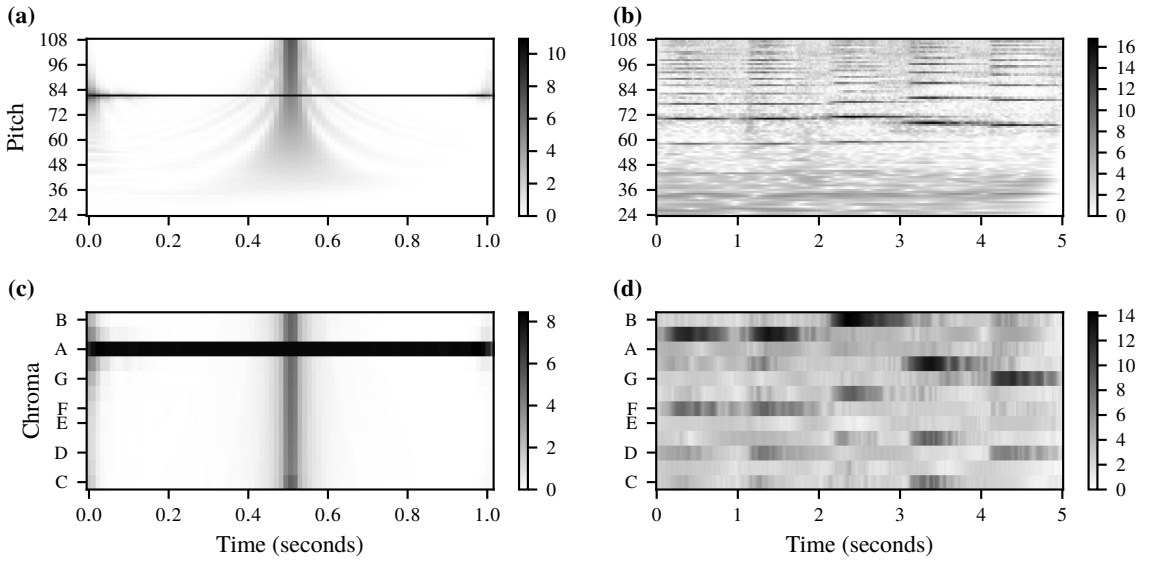


Figure 2.7: IIR-based visualizations ($N = 512$, $H = 256$). **(a)** Spectrogram for `Synt`. **(b)** Spectrogram for `Beet`. **(c)** Chromagram for `Synt`. **(d)** Chromagram for `Beet`.

Figures 2.7a and b show visualizations of IIR-based transforms of our examples using a rectangular window with a size of about 23.2 ms and a hop size of about 11.6 ms (512 and 256 samples in a sampling rate $F_s = 22\,050$ Hz, respectively). In Figure 2.7a, we clearly see the sinusoidal component with 880 Hz ($p = 81$) without energy in the neighboring bins (no spectral leakage).[5] As another observation, there is a temporal smoothing in the lower frequency range, which leads to a spreading of the impulse in that range. Figures 2.7c and d show corresponding chroma features, which are computed by a chroma reduction procedure as described in Section 2.3. Compared to the STFT-based chroma representations (Figure 2.5), there is no spectral leakage visible in the chromagrams. In Figure 2.7c, we now see the chroma A of the sinusoid without substantial energy in neighboring bands. In Figure 2.7d, we can clearly follow the pitch class sequence ($B^\flat$, $B^\flat$, B, $A^\flat$, G) of the music excerpt (Figure 2.2b).

2.5 Constant-Q Transform and Harmonic Constant-Q Transform

Another time–frequency representation having logarithmically spaced frequency values is the constant-Q transform (CQT), originally introduced by Brown [29]. An efficient algorithm and implementation for computing the CQT, widely used in MIR research, was proposed by Schörkhuber and Klapuri [171].[6] There are three key differences between the IIR-based transform and the CQT. First, the CQT is based on FIR filters instead of IIR filters. Second, it uses different window sizes that depend on the frequency band (larger window sizes for lower frequencies). As a result, the frequency resolution is higher for low

[5] Note that the zero-padding strategy described above may lead to boundary effects at the beginning and end of the signal, similar to spectral leakage.

[6] We also want to reference the technical report [24], which is not a formal publication, but helpful from a practical perspective for understanding the CQT.

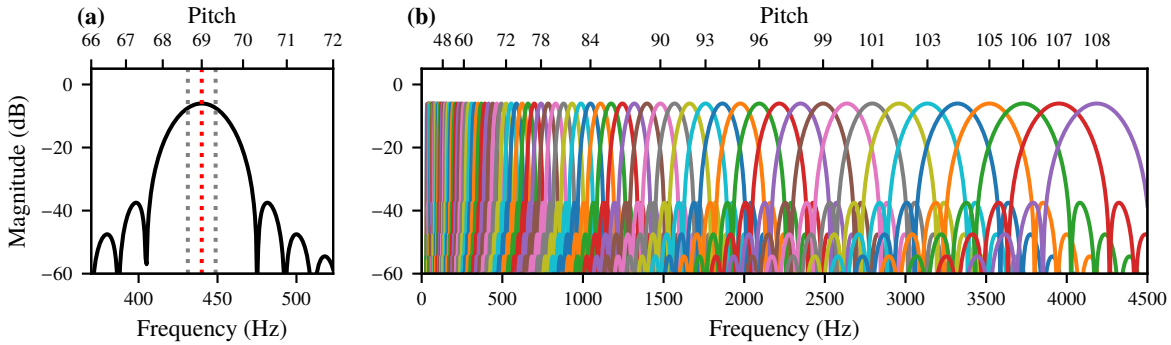


Figure 2.8: **(a)** Frequency response of a CQT bandpass filter with a center frequency of 440 Hz (indicated with dotted red line) and a Q factor of 25 (resulting bandwidth of 17.6 Hz indicated with dotted gray lines). **(b)** Frequency responses for the filterbank used in the CQT.

frequencies, and the time resolution is higher for high frequencies. Third, the transform is invertible and, thus, allows the reconstruction of the time-domain signal from the time–frequency domain.

Recall that the Q factor $Q \in \mathbb{R}$ of a bandpass filter is the quotient of the center frequency to the bandwidth (as described in the previous section). A constant Q factor means that the passband has a constant width on a logarithmic frequency axis—which is desirable from a musical perspective. Both the IIR-based transform of Section 2.4 and the CQT described in this section use filters with a constant Q factor. However, the name *constant-Q transform* is widely used in the MIR community for the specific transform of this section.

The CQT $\mathcal{X}^{\mathrm{CQT}}$ of a signal x is given by

$$\mathcal{X}^{\mathrm{CQT}}(m,k) = \sum_{n=0}^{N_k-1} x(n+mH_k)\overline{a_k(n)}, \tag{2.9}$$

where $m \in \mathbb{Z}$ denotes a frame index, and $k \in [0 : K-1]$ denotes a pitch index corresponding to the frequency $f(p_k)$ (see Eq. 2.5). $N_k \in \mathbb{N}$ is a frequency-dependent window size, which increases with decreasing frequency: $N_k = \lceil Q \cdot F_\mathrm{s}/f(p_k) \rceil$. $H_k \in \mathbb{N}$ is a frequency-dependent hop size, which may also increase with decreasing frequency, or, alternatively, may be constant, which leads to the same number of frames in all frequency bins (i.e., a matrix shape of the CQT). In the following, we use a constant hop size. $\overline{a_k(n)}$ denotes the complex conjugate of the so-called time–frequency atom

$$a_k(n) = \frac{1}{N_k} w_k(n) \exp\left(2\pi i n f(p_k)/F_\mathrm{s}\right), \tag{2.10}$$

for $n \in [0 : N_k - 1]$, used to analyze the signal x with respect to the frequency $f(p_k)$. Here, the window $w_k : [0 : N_k - 1] \to \mathbb{R}$ is similar to the window used in the STFT. For more details, we refer to the original literature [29, 171].

We can interpret the set of time–frequency atoms as an FIR filterbank. Figure 2.8a shows the frequency response of a time–frequency atom for $f(p_k) = 440$ Hz. Here, settings similar to those used for the

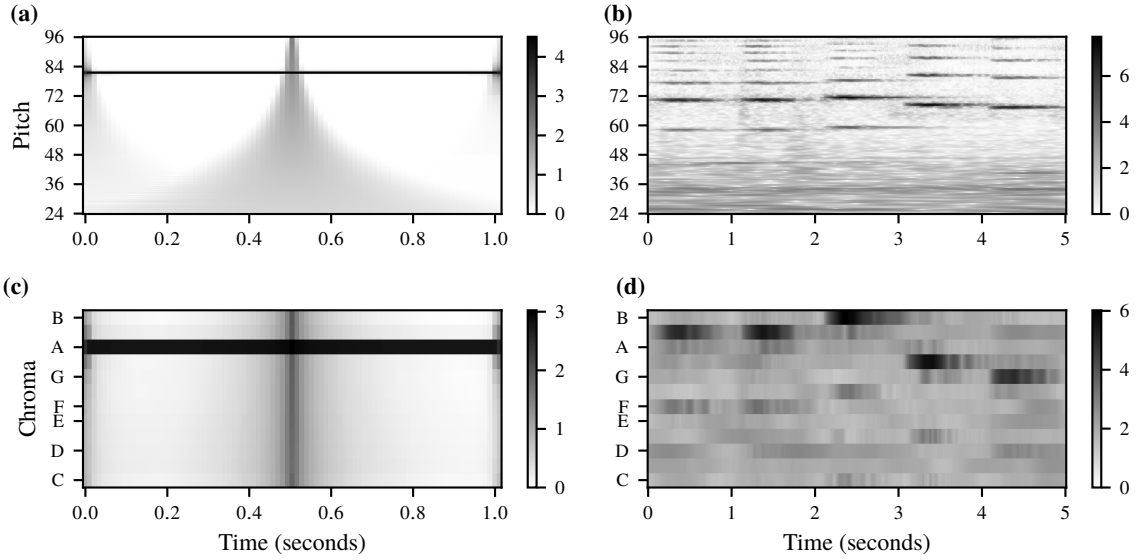


Figure 2.9: CQT-based visualizations ($H = 256$). **(a)** Spectrogram for `Synt`. **(b)** Spectrogram for `Beet`. **(c)** Chromagram for `Synt`. **(d)** Chromagram for `Beet`.

frequency responses in the IIR-based transform (Figure 2.6) are used (i.e., $Q = 25$, $B = 12$ bins per octave, $K = 85$ pitch bins). Compared to the frequency response of the IIR-based transform, the transition bands are wider, and the side lobes have a higher amplitude. The same findings can also be verified for the frequency responses of the entire filter bank of the CQT (see Figure 2.8b).

We described a naive method to compute the CQT, which we also used to derive the frequency responses of the time–frequency atoms in Figure 2.8. However, in practice, we use a computationally efficient implementation for computing the CQT, based on the FFT and a frequency-domain kernel. Furthermore, a multi-resolution procedure is used, where the input signal is downsampled to different sampling rates, and the time–frequency atoms are reused at different rates. For more details, we refer to the paper by Schörkhuber and Klapuri [171].

Figure 2.9 shows visualizations of CQTs for our examples. We find various similarities to the visualizations for the IIR-based transform (Figure 2.7). In Figure 2.9a, we also find some temporal smearing in the lower frequency bands. Figures 2.9a and b clearly show the sinusoid having a pitch of 81 and a chroma of A, respectively. There is no spectral leakage visible in the chromagrams. In Figure 2.9d, we can identify the pitch class sequence of our musical example. In general, the figures show that the CQT- and the IIR-based transforms lead to similar results.

Next, we want to point out some connections between the CQT and the IIR-based transform from a practical perspective. Both transforms have similar parameters, such as the lowest frequency considered $f(p_0) \in \mathbb{R}$, the number of bins per octave $B \in \mathbb{N}$, the number of pitch bins $K \in \mathbb{N}$, the Q factor $Q \in \mathbb{R}$, and the type of the window function w. For the figures of this chapter, we used identical parameters for both time–frequency transforms (if not specified otherwise). However, in practice, the transforms are typically

Table 2.1: Typical parameters used for the CQT and the IIR-based transform.

Parameter	**IIR**	**CQT**
Lowest frequency $f(p_0) \in \mathbb{R}$	$f(24) \approx 32.703$	32.7
Bins per octave $B \in \mathbb{N}$	12	36 / 60
Number of pitch bins $K \in \mathbb{N}$	85	216 / 360
Q factor $Q \in \mathbb{R}$	25	$(2^{1/B} - 1)^{-1}$
Window function w	Rectangular	Hann

associated with distinct parameter settings, summarized in Table 2.1. Some of the differences are due to common practices and the default settings in the implementations used[7], rather than to conceptual reasons. For example, according to the default settings in the implementations used, the lowest considered frequency (corresponding to C_1) is 32.7 (rounded to the first decimal place) in the CQT and $f(24) \approx 32.703$ (without rounding) in the IIR-based transform, which leads to a "tuning difference" of about 0.17 cents. Another difference is the setting of the Q factor. In the CQT, one may expect a higher Q factor given the frequency responses' wide transition bands. A higher Q factor would decrease the filters' bandwidth, which may be desirable to avoid frequency smearing. However, for high Q factors, one may not be able to perfectly reconstruct the time-domain signal from a CQT [171], which also is a desired property. To compensate for the potential frequency smearing, one typically uses more than 12 semitones per octave (e.g., 36 or 60). In the IIR-based transform, which is not perfectly reconstructable anyway, the transition bands are narrower. Another difference between the transforms is the type of window function typically used. In the CQT, a smooth, "bell-shaped" window (e.g., a Hann window) is used instead of a rectangular window to avoid hard edges in the windowed complex exponentials. The window is an essential part of the time–frequency atom (and is also part of the FIR filter). In the IIR-based transform, the window is applied after filtering (being not part of the IIR filter) for the short-time energy computation. Here, there is no need for soft edges in the window function. Thus, the usage of rectangular windows is not problematic in the IIR-based transform. For the spectrogram visualizations in this chapter, as well as throughout the rest of this thesis, we use the transforms with the parameter settings listed in Table 2.1, if not mentioned otherwise.

The CQT is also the basis for another audio representation named harmonic CQT (HCQT) that was designed as an input to a convolutional neural network (CNN) [23]. The CQT is a matrix of dimensions time and pitch. In contrast, the HCQT is a three-dimensional tensor of dimensions time, pitch, and harmonics. Let us recall that $f(p_0) \in \mathbb{R}$ is the lowest frequency considered in the CQT, also referred to as base frequency. In the HCQT, we stack several CQT matrices with different base frequencies, where the frequencies are chosen to form a harmonic series. For a given frequency $f(p_0)$, the CQT matrices of the HCQT have a base frequency $h \cdot f(p_0)$, where $h \in \{0.5, 1, 2, 3, 4, 5\}$. As a consequence, all time–frequency bins across the depth of the HCQT tensor are harmonically related. Thus, the convolutional kernels of a CNN can easily exploit these harmonic relationships. The subharmonic frequency $h = 0.5$ is additionally chosen for the first HCQT slice, even if it is not part of the harmonic series. It is added for helping to disambiguate between the fundamental frequency and the first harmonic [23].

[7] We use the implementations from librosa 0.8.0 [119].

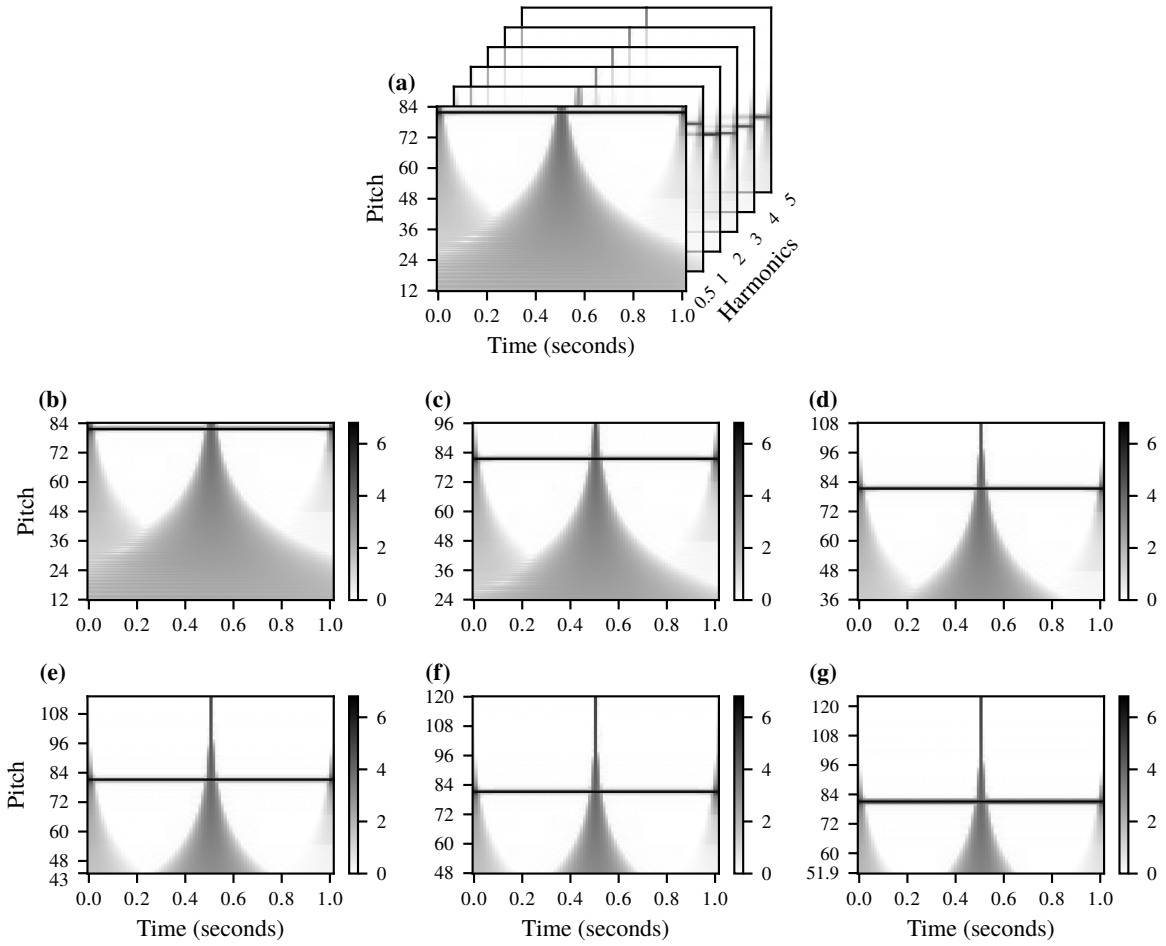


Figure 2.10: Visualizations of the HCQT for `Synt`. **(a)** Entire tensor. **(b)** Slice with $h = 0.5$, **(c)** $h = 1$, **(d)** $h = 2$, **(e)** $h = 3$, **(f)** $h = 4$, and **(g)** $h = 5$.

Figure 2.10a illustrates an HCQT tensor for our synthetic example. The front matrix is the HCQT slice corresponding to $h = 0.5$. In this slice, we have the lowest pitch range, and the sinusoid (horizontal line at $p = 81$) is barely contained in this range. Figures 2.10b–g show all slices of the tensor separately. Note the different range of the vertical axis in each slice due to the changing base frequencies. As a consequence, the horizontal line appears at different vertical positions in each slice. Due to the harmonic relations of the base frequencies, the lowest frequency of a slice does not need to correspond to an integer pitch p. For example, using $f(p_0) = 32.7\,\text{Hz}$, the base frequency for $h = 5$ is $5 \cdot 32.7 = 163.5\,\text{Hz}$, which is slightly below the equal-tempered pitch of E_3 corresponding to $p = 52$. Instead, it corresponds to E_3 in just intonation.

Part I

Learning Low-Dimensional Shingle Embeddings

3 Learning Low-Dimensional Embeddings of Audio Shingles for Cross-Version Music Retrieval

This chapter is based on [216]. The first author Frank Zalkow is the main contributor to this article. In collaboration with his supervisor Meinard Müller, he formalized the problem, developed the ideas, and wrote the paper. Furthermore, Frank Zalkow implemented the approaches and conducted the experiments.

Cross-version music retrieval aims at identifying all versions of a given musical piece using a fragment of a music representation as a query, e.g., a short audio snippet. One previous approach, which is particularly suited for Western classical music, is based on a nearest neighbor search using short sequences of chroma features, also referred to as audio shingles. From the viewpoint of efficiency, dimensionality reduction and indexing are important aspects. In this chapter, we extend previous work by adapting two embedding techniques; one is based on classical principle component analysis, and the other is based on neural networks trained with the triplet loss. Furthermore, we report on systematically conducted experiments with Western classical music recordings and discuss the trade-off between retrieval quality and embedding dimensionality. As one main result, we show that, using neural networks, one can reduce the audio shingles from 240 to fewer than 8 dimensions with only a moderate loss in retrieval accuracy. In addition, we present extended experiments with databases of different sizes and different query lengths to test the scalability and generalizability of the dimensionality reduction methods. We also provide a more detailed view into the retrieval problem by analyzing the distances that appear in the nearest neighbor search.

3.1 Introduction

Large amounts of digitally available music data require efficient retrieval strategies. In recent decades, many systems for music retrieval based on the query-by-example paradigm have been suggested. Given a fragment of a music representation as a query, the task is to automatically retrieve documents from a music database containing parts or aspects that are similar to the query [40, 73, 194]. One such retrieval scenario is known as audio identification or fingerprinting [37, 41, 128, 202], where the user specifies a

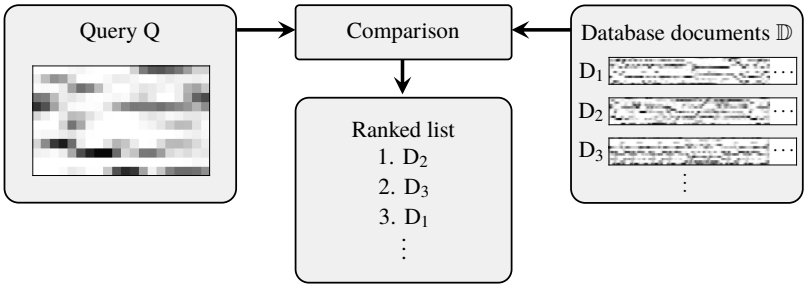


Figure 3.1: Overview of the retrieval procedure: A query (Q) is compared with a set 𝔻 of database documents, resulting in a ranked list of documents.

query using an excerpt of an audio recording, and the task is to identify the particular audio recording that is the source of the query. A more challenging scenario is cross-version retrieval, including tasks such as audio matching, version identification, or cover song retrieval [39, 56, 99, 107, 122, 175, 177]. Here, given an excerpt of an audio recording as a query, the goal is to automatically retrieve all recordings in a database that correspond to the same piece of music as the query. Relevant documents may include various interpretations, arrangements, and cover songs of the piece underlying the recording of the query fragment. We focus on such a retrieval scenario in the context of Western classical music, where one typically has many different performances (referred to as versions) for the same piece of music.

For example, given a 10- to 30-second fragment of a recording of Beethoven's Fifth Symphony performed by the Berlin Philharmonic conducted by Karajan, the task is to identify all versions of this symphony in a database, including an interpretation by the New York Philharmonic conducted by Bernstein and an interpretation by the Vienna Philharmonic conducted by Abbado.

Figure 3.1 illustrates a typical retrieval procedure, where query and database recordings are compared employing chroma-based audio representations [128, Chapter 3], resulting in a ranked list of database documents. For comparison, a temporal alignment procedure (e.g., SDTW [128, Chapter 7]) is often used to compensate for non-linear tempo differences between the query and relevant database documents [166, 175]. However, for huge data collections, the resulting runtime of such approaches is prohibitive. As a more efficient alternative, previous work [39, 40, 72] introduced shingling approaches, where short feature sequences are used for indexing. In this chapter, we build on a study presented by Grosche and Müller [72], who approached this task using chroma-based audio shingles (see the left part of Figure 3.1 for a visualization of such a shingle). Retrieval was performed via locality-sensitive hashing (LSH) applied to entire shingles. LSH is a random indexing technique for approximate nearest neighbor search [183]. The authors investigated the feature design, the length of the shingles, and the effect of dimensionality reduction applied to *individual* feature vectors. In this chapter, we propose approaches to increase the efficiency of the retrieval even more. We concentrate on the aspect of dimensionality reduction applied to *entire* shingles so that standard tree-based indexing techniques for nearest neighbor search can be used for retrieval.

As one contribution of this chapter, we first use an approach based on principal component analysis (PCA) applied to entire shingles, rather than to individual chroma vectors as in previous work [72]. As another

contribution, we then adapt CNNs trained with the triplet loss [173] to further reduce the shingles' dimensionality without losing their discriminative power. We conduct basic experiments with a medium-sized collection of music recordings to study the benefits and limitations of the dimensionality reduction methods. As our main result, we show that the shingle dimension can be reduced from 240 to below 8 with only a moderate loss in retrieval quality. Furthermore, we report on extended experiments with a larger dataset, using different query lengths to study the scalability and generalizability of the embedding approaches. In our context, scalability refers to the dataset size, and generalizability refers to the diversity of the dataset. We also provide detailed insights into the challenges of the retrieval task and their musical reasons by analyzing the distance distributions that form the basis for the nearest neighbor search.

The structure of this chapter is as follows. We give an overview of related work in Section 3.2 and formalize our retrieval scenario in Section 3.3. We describe our embedding approaches in Section 3.4. Then, in Section 3.5, we report on our basic experiments based on a medium-sized dataset. Finally, in Section 3.6, we provide further insights based on our extended experiments using a larger and more diverse dataset, and conclude in Section 3.7 with a short summary.

3.2 Related Work

On a rough level, we can categorize music retrieval scenarios into metadata- and content-based retrieval tasks [40]. Metadata-based systems use textual information for searching in music databases. On the contrary, content-based strategies use actual music data such as sheet music images, symbolic music representations, or audio data. Content-based systems can be further categorized according to the modalities involved. For an overview of multimodal music retrieval scenarios, we refer to a survey by Müller et al. [136]. In this chapter, we focus on retrieval scenarios, where both query and database documents are audio recordings. In such a setting, we have a query that is either a segment of a music recording or a complete recording. The goal is then to retrieve music recordings that are similar to the query, based on some notion of similarity. Following Casey et al. [40] and Grosche et al. [73], we can categorize such retrieval scenarios according to two properties: specificity and granularity. Specificity refers to the degree of similarity between the query and the database documents. High specificity is related to a strict notion of similarity, whereas low specificity refers to a rather vague one. The granularity refers to the length of the query, ranging from a short audio snippet (a couple of seconds) to an entire recording (several minutes).

A typical task of high specificity and low granularity is audio identification or audio fingerprinting, where the task is to identify the particular audio recording that is the source of the query [38, 202]. At the lower end of the specificity scale are tasks such as genre recognition [195]. A medium-level specificity is associated with tasks such as audio matching [72, 99], version identification [149, 166], live song detection [154, 193], and cover song retrieval [30, 42, 84, 100, 105, 138, 169, 174, 178, 192, 209]. In all these tasks, one allows for variations as they typically occur in different performances and arrangements of

a piece of music. The tasks differ in their granularity (e.g., shorter queries for audio matching and longer queries for version identification) and the specific types of music recordings of interest (e.g., live versions by the same performers for live song detection, or popular music with different performers for cover song retrieval). Cover song retrieval is a well-established research task, where one considers variations as they occur in different performances of the same piece of popular music. Such variations concern many different musical facets, including timbre, tempo, timing, structure, key, harmony, and lyrics [177]. We denote the set of all considered versions of a particular piece of music as "clique." In the case of cover song retrieval, such a clique may consist of the original version of a song (such as "Hallelujah" by Leonard Cohen) and all cover versions of the same song (by other artists, e.g., Jeff Buckley). A task that is similar to cover song retrieval is version identification for Western classical music, where one allows for variations as they occur in different performances of the same piece of Western classical music [8, 72, 122, 124]. One clique in this task, for example, may consist of all performances of Beethoven's Fifth symphony. This scenario is associated with a higher specificity than cover song retrieval because we expect fewer variations in Western classical music than in popular music. For example, the rough harmonic progression is the same among different performances of the same classical piece, which is not always the case for cover songs of popular music. For that reason, cover song retrieval can be considered a more difficult task compared to version identification for classical music. For more details on this, we refer to the overview article by Serrà et al. [177]. In this chapter, we focus on audio matching or version identification for Western classical music.

Miotto and Orio address version identification for classical music by modeling each musical work with a hidden Markov model (HMM) [122, 123]. In these studies, a query is identified by choosing the HMM that models the query with the highest probability. To avoid the time-consuming evaluation of all HMMs, the authors propose first to select a small subset of potential candidates [123] and then to evaluate only the HMMs for the most promising candidates. Instead of HMMs, other audio alignment algorithms such as particle filtering, have been used in similar settings [124]. Classical music retrieval was also approached as a multimodal scenario [7, 8], in particular, using audio and symbolic representations [153]. Arzt et al. [7, 8] present an approach that uses symbolic music representations (as the database) to identify a query audio snippet of classical piano music. The query is first transcribed into a series of symbolic events by a neural network. Then, a symbolic fingerprinting algorithm can be applied. This system has a good performance for music where the automatic transcription step achieves good results, e.g., piano music. However, the approach is problematic for kinds of music, where automatic transcription is more challenging, such as complex orchestral music. Another line of work uses chroma feature sequences of short audio fragments for audio matching of classical music [72, 99]. Our contribution builds upon this line of work, and we refer to these studies in the following sections.

In recent years, several authors applied neural networks for computing features that are then used for version retrieval tasks. The main focus of these works is cover song retrieval of popular music. In some studies, the features are obtained using autoencoders that are trained to reconstruct their input representations, e.g., chromagrams [59, 214]. In other studies, the network is trained similarly to training

procedures for classification problems, where the aim is to assign a class label (among a finite set of labels) to an input recording [87, 208, 212, 213]. Here, the classes correspond to the number of cover groups (or cliques) of the training set. For the retrieval application, one may only consider this limited number of groups and directly use the network's classification for the recordings of the test set. Otherwise, if cover groups are considered that are not included in the training set, one uses an intermediate representation of the network (the feature map before the classification layer) as an audio representation for the retrieval task. As another training paradigm, siamese neural networks are applied in several studies, where the network computes features for two input recordings. The network's features for the input recordings are then compared with each other to derive a loss used for training. The comparison can be made on the basis of binary cross-entropy [184] or dedicated loss functions for representation learning, such as the triplet loss [48, 49, 50, 149, 211]. This training procedure is usually applied to obtain fixed-size vectors that represent entire audio recordings. Often these vectors are high-dimensional (e.g., 64 to 1024 as in [48] or even 16 000 as in [211]). A subsequent reduction of these high-dimensional vectors has proven to be beneficial for the retrieval results [210]. In the approach presented in this chapter, we also apply the triplet loss for version retrieval (more details in Section 3.4.3). In our approach, there are three key differences to other studies from the literature using the triplet loss for this task. First, we target Western classical music rather than popular music. Second, we directly obtain low-dimensional vectors of a size below 30. Third, these vectors represent local sections (shingles) of an audio recording instead of entire recordings.

3.3 Shingle-Based Retrieval Scenario

Closely following Grosche and Müller [72], we now formalize the shingle-based retrieval strategy used in this chapter. Given a short fragment of a music recording as a query, the goal is to retrieve all versions (documents) of the same piece of music underlying the query. To this end, we compare the database and query recordings based on a particular feature representation. The retrieval result for a query is given as a ranked list of documents. Figure 3.1 illustrates this general procedure. In the following, we explain the feature computation as well as the retrieval approach.

Our approach is based on so-called "shingles" [39], which are short sequences of feature vectors. We denote such a shingle of feature dimension $F \in \mathbb{N}$ and length $L \in \mathbb{N}$ by $\boldsymbol{S} \in \mathbb{R}^{F \times L}$. In general, we generate such shingles from audio recordings, which are represented by longer feature sequences of variable length. The feature sequence of an audio recording is denoted by $\boldsymbol{C} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N)$ of length $N \in \mathbb{N}$ and consists of feature vectors $\boldsymbol{c}_n \in \mathbb{R}^F$ for $n \in \{1, \ldots, N\}$. We use chroma-based audio features, which measure local energy distributions of the audio recording in the $F = 12$ chromatic pitch class bands [128, Chapter 3] (computed on the basis of a log-frequency spectrogram, as described in Section 2.4). More precisely, we use a variant called CENS (chroma energy distribution normalized statistics) [132], which are chroma features that are processed using a procedure to make them better suited for retrieval: First, each chroma vector is ℓ^1-normalized. Then, the resulting values of the chroma features are quantized in a logarithmic

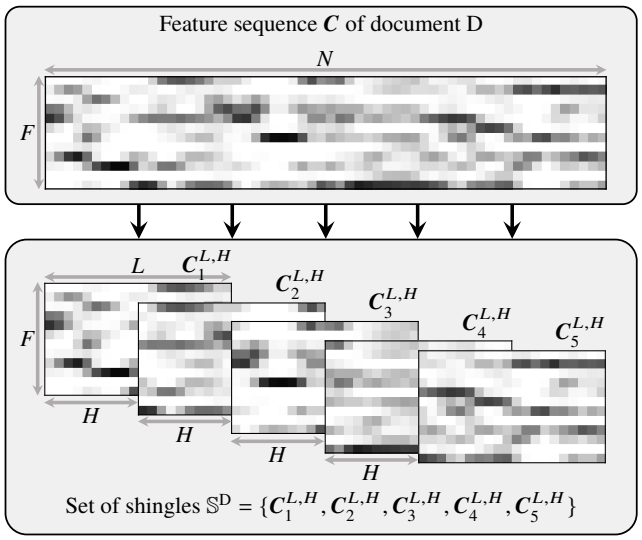


Figure 3.2: Schematic overview of the shingle generation process: A set of overlapping shingles $\mathbb{S}^{\mathrm{D}}$ is generated from the feature sequence of document D.

way (by mapping logarithmically spaced value ranges to integer values, e.g., values between 0.05 and 0.1 are mapped to 1, values between 0.1 and 0.2 to 2, etc.). Next, the chroma feature sequence is temporally smoothed (using a smoothing length of 4 seconds) and downsampled (from 10 Hz to 1 Hz). Finally, each chroma vector is ℓ^2-normalized. The most important aspect of this post-processing procedure is the temporal smoothing because it makes the features more robust against tempo differences. This chroma variant is state-of-the-art for the given task [72]. The upper part of Figure 3.2 shows a visualization of the feature type used in this chapter. To generate shingles of length $L < N$ from the feature sequence $\boldsymbol{C}$, we use a hop size $H \in \mathbb{N}$ to define the subsequences

$$\boldsymbol{C}_n^{L,H} := (\boldsymbol{c}_{(n-1)H+1}, \ldots, \boldsymbol{c}_{(n-1)H+L}) \tag{3.1}$$

for $n \in \{1, \ldots, \lfloor \frac{N-L}{H} \rfloor + 1\}$, where $\lfloor \cdot \rfloor$ denotes the floor operation. The resulting subsequences can also be regarded as matrices or shingles $\boldsymbol{C}_n^{L,H} \in \mathbb{R}^{F \times L}$. For brevity, we will omit the superscript H in the case of $H = 1$. See the lower part of Figure 3.2 for such a sequence of overlapping shingles.

We now describe the retrieval approach for a fixed query (denoted as Q) and database document (denoted as D). For now, the query consists of a single shingle $\boldsymbol{S}^{\mathrm{Q}} \in \mathbb{R}^{F \times L}$. Previous investigations of the query length found that a length of 20 seconds is well-suited for performing the retrieval with a single audio shingle [72]. In our study, we use such a shingle length, resulting in a shingle dimensionality of $F \cdot L = 12 \cdot 20 = 240$. The document D is represented by a set of shingles

$$\mathbb{S}^{\mathrm{D}} = \left\{\boldsymbol{C}_n^L : n \in \{1, \ldots, N - L + 1\}\right\}. \tag{3.2}$$

This set $\mathbb{S}^{\mathrm{D}}$ consists of all subsequences $\boldsymbol{C}_n^L$ from the audio recording of document D (as defined in Equation 3.1), generated with a hop size $H = 1$. In the next step, $\boldsymbol{S}^{\mathrm{Q}}$ is compared with all shingles from the set $\mathbb{S}^{\mathrm{D}}$. The comparison between Q and D is achieved by first transforming the shingles to vectors by a

function

$$f\colon \mathbb{R}^{F\times L} \to \mathbb{R}^{K} \tag{3.3}$$

for some $K \in \mathbb{N}$. In the brute-force case, f just flattens a matrix by concatenating all columns (i.e., $K = F \cdot L$). Using shingle embedding methods, as explained later, f performs a dimensionality reduction (typically $K \ll F \cdot L$). Given two shingles $\boldsymbol{S}^{(1)}$ and $\boldsymbol{S}^{(2)}$, we compare them in the embedding space using a distance function

$$d\colon \mathbb{R}^{K} \times \mathbb{R}^{K} \to \mathbb{R}_{\geq 0}. \tag{3.4}$$

In the following, we use the squared Euclidean distance

$$d(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \sum_{k=1}^{K} (x_k^{(1)} - x_k^{(2)})^2, \tag{3.5}$$

where $\boldsymbol{x}^{(1)} = f(\boldsymbol{S}^{(1)})$ and $\boldsymbol{x}^{(2)} = f(\boldsymbol{S}^{(2)})$. Given a query Q (in the form of a shingle $\boldsymbol{S}^{\mathrm{Q}}$) and a database document D (in the form of a set of shingles $\mathbb{S}^{\mathrm{D}}$), we compute the distance between Q and D by

$$\delta_{\mathrm{D}}(\boldsymbol{S}^{\mathrm{Q}}) \coloneqq \min_{\boldsymbol{S}\in\mathbb{S}^{\mathrm{D}}} d\big(f(\boldsymbol{S}^{\mathrm{Q}}), f(\boldsymbol{S})\big). \tag{3.6}$$

Finally, for Q and a data collection $\mathbb{D}$ containing $|\mathbb{D}|$ documents, we compute $\delta_{\mathrm{D}}(\boldsymbol{S}^{\mathrm{Q}})$ between Q and all $\mathrm{D} \in \mathbb{D}$ and rank the results by ascending $\delta_{\mathrm{D}}(\boldsymbol{S}^{\mathrm{Q}})$.

Previous work [72] used maximum cosine similarity instead of minimum Euclidean distance. We use the squared Euclidean distance because this distance naturally occurs in both dimensionality reduction methods, as explained in Section 3.4. Note that, since each feature vector of the shingles is normalized, Euclidean distance and cosine similarity lead to similar retrieval results.[8]

3.4 Shingle Embedding

In this section, we explain the brute-force approach and the two shingle embedding techniques. The first uses standard PCA, and the second is based on siamese neural networks, which are trained with the so-called triplet loss.

3.4.1 Brute Force

As a baseline approach, we just flatten each audio shingle $\boldsymbol{S} \in \mathbb{R}^{12\times 20}$ to a vector of size $K = 12 \cdot 20 = 240$ by concatenating the feature vectors. In other words, we do not apply any dimensionality reduction. Note that no training is involved in this approach, contrary to the embedding methods explained in the following

[8] The relation between the squared Euclidean distance of ℓ^2-normalized vectors and their cosine similarity is given by: $d(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = 2\big(1 - \cos(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)})\big)$.

subsections. This baseline approach of using the full-dimensional audio shingles was also used in the study by Grosche and Müller [72].

3.4.2 PCA

As a first embedding approach, we use PCA [21] to reduce the dimensionality of the audio shingles. Each audio shingle $\boldsymbol{S} \in \mathbb{R}^{12\times 20}$ is seen as a vector of size $12 \cdot 20 = 240$. PCA learns an orthonormal basis that is used to project these audio shingles onto a lower-dimensional linear subspace. Using a given training set, this basis is learned in order to minimize the average projection cost, defined as the squared Euclidean distance between the original shingles and their projections [21]. Dimensionality reduction is then performed by considering only the first basis vectors instead of the complete set of basis vectors. As a result, one obtains a linear transformation that maps an audio shingle $\boldsymbol{S}$ to an embedding vector $\boldsymbol{x} \in \mathbb{R}^K$. This mapping $f(\boldsymbol{S}) = \boldsymbol{x}$ is then used in the experiments with a separate test set by embedding the entire database as well as the queries before performing the retrieval (see Section 3.5.4). The authors of [72] also used PCA for dimensionality reduction, but they applied PCA only to individual chroma vectors of dimension F. Thus, their approach can only make use of the chroma dimension, without any temporal information. In our approach, we apply PCA to entire shingles of dimension $F \cdot L$ and can, therefore, exploit redundancies in the temporal sequence of chroma vectors for dimensionality reduction.

3.4.3 Neural Network with Triplet Loss

As a second embedding approach, we use neural networks to reduce the dimensionality of the audio shingles. Such networks can be used to learn non-linear functions that map high-dimensional input representations to low-dimensional output representations.

For learning embeddings, a loss function is used that enforces specified similarities and dissimilarities in the embedding space. An example of such a loss function is the contrastive loss [75], which has been used, e.g., for audio-to-score alignment [4]. Another example is the hinge loss [65, 90], which has been used, e.g., to learn representations for cross-modal score-to-audio embeddings [52] and artist similarity [139]. A related loss function is the triplet loss, which was developed for the task of face recognition [173], and then adapted for audio and music processing tasks such as speech retrieval [77], sound event classification [86], audio fingerprinting [5], artist clustering [163], music similarity [111], and cover song retrieval [49, 149, 211]. We now show how to reduce the audio shingles' dimensionality for our cross-version music retrieval scenario by employing a CNN trained with the triplet loss. During training, our network embeds three shingles for computing the loss, which is then used to update the network's parameters.

Figure 3.3 shows an illustration of the triplet embedding. To define the triplet loss in our scenario, we select three shingles: An anchor shingle $\boldsymbol{S}^{\mathrm{a}}$, a positive shingle $\boldsymbol{S}^{\mathrm{p}}$, and a negative shingle $\boldsymbol{S}^{\mathrm{n}}$ (see the upper

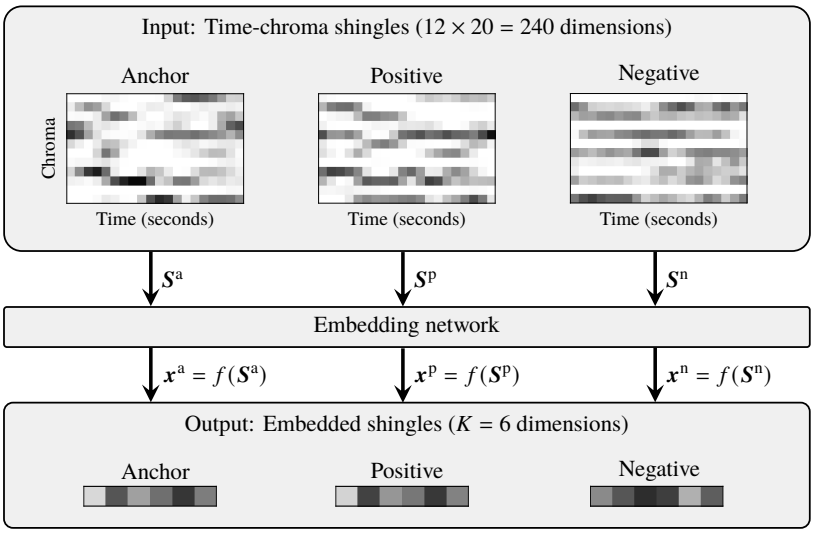


Figure 3.3: Schematic overview of the triplet embedding: Anchor, positive, and negative shingles $(\boldsymbol{S}^a, \boldsymbol{S}^p, \boldsymbol{S}^n)$ are embedded into the vectors $(\boldsymbol{x}^a, \boldsymbol{x}^p, \boldsymbol{x}^n)$ by a neural network.

part of Figure 3.3 for an example). With respect to the anchor shingle, the positive shingle is musically similar, while the negative shingle is musically dissimilar. More precisely, the anchor and positive shingles originate from different versions of the same piece of music and correspond to the same musical position within that piece. Anchor and negative shingles do not correspond to each other.

The goal is to find an embedding function $f\colon\ \mathbb{R}^{F\times L} \to \mathbb{R}^K$ such that $f(\boldsymbol{S}^a)$ is numerically close to $f(\boldsymbol{S}^p)$ and far from $f(\boldsymbol{S}^n)$. The lower part of Figure 3.3 visualizes embeddings with this property. Let

$$\boldsymbol{X} = (\boldsymbol{x}^a, \boldsymbol{x}^p, \boldsymbol{x}^n) = \big(f(\boldsymbol{S}^a), f(\boldsymbol{S}^p), f(\boldsymbol{S}^n)\big). \tag{3.7}$$

Then, following Schroff et al. [173], we define the loss as

$$\mathcal{L}(\boldsymbol{X}) = \max\big(0, d(\boldsymbol{x}^a, \boldsymbol{x}^p) - d(\boldsymbol{x}^a, \boldsymbol{x}^n) + \alpha\big), \tag{3.8}$$

with d being a distance function and $\alpha \in \mathbb{R}_{\geq 0}$ being a margin parameter. This parameter α specifies the minimum difference between the distance of the anchor–positive pair and the distance of the anchor–negative pair that we aim for in the embedding space. As for the distance function d, we use the squared Euclidean distance in our experiments, as defined in Equation 3.5. The cost function J to be optimized during mini-batch gradient descent is the average of losses over a mini batch $\mathbb{B}$ of triplets:

$$J(\mathbb{B}) = \frac{1}{|\mathbb{B}|} \sum_{\boldsymbol{X}\in\mathbb{B}} \mathcal{L}(\boldsymbol{X}). \tag{3.9}$$

Our neural network architecture is summarized in Table 3.1. It comprises two blocks, each consisting of two convolutional layers and a max-pooling layer. Finally, a dense layer reduces the network's internal representation to the embedding dimensionality K and an ℓ^2-normalization layer ensures that the embedding vectors do not become arbitrarily large or small. This topology is inspired by a network for

Layer	Output Shape	Parameters
Input	(12, 20, 1)	
Conv2D* 12×(3, 3, 1)	(12, 20, 12)	108
BatchNorm + ELU	(12, 20, 12)	48
Conv2D 12×(3, 3, 12)	(12, 20, 12)	1296
BatchNorm + ELU	(12, 20, 12)	48
MaxPool2D (2, 2)	(6, 10, 12)	
Conv2D 12×(3, 3, 12)	(6, 10, 12)	1296
BatchNorm + ELU	(6, 10, 12)	48
Conv2D 12×(3, 3, 12)	(6, 10, 12)	1296
BatchNorm + ELU	(6, 10, 12)	48
MaxPool2D (2, 2)	(3, 5, 12)	
Flatten + Dense K	(K)	$K \cdot 181$
Output: ℓ^2-normalize	(K)	

Table 3.1: Neural network architecture. For all convolutional operations, we use a zero-padding size of 1. Conv2D* indicates a circular padding of the chroma axis instead of zero padding.

multimodal music embeddings [52]. We simplified the architecture (e.g., using 2 blocks instead of 4) for two reasons: First, our scenario is monomodal and therefore less complex compared to the multimodal task [52], and second, our input dimension is smaller (using chroma features instead of spectral features). Our network has relatively few parameters compared with many other deep-learning systems. E.g., for $K = 10$, the network has about 6000 parameters.

3.5 Basic Experiments

In this section, we report on our experiments with medium-sized datasets for training and testing. Later, in Section 3.6, we will also report on experiments with an extended dataset. For now, we use medium-sized datasets similar to those used in a previous study [72]. First, we describe the datasets and our evaluation measures. Second, we discuss the evaluation results obtained using the brute-force approach, the PCA-based embeddings, and the approach using neural networks. Third, we analyze the influence of the margin parameter α (used in the loss function) on the retrieval results. Finally, we report on a runtime experiment that indicates the impact of the embeddings' dimensionality on the retrieval time.

3.5.1 Training and Testing Datasets

In our experiments, we use audio recordings of Western classical music. In particular, we use pieces from three composers: Symphonies by Beethoven, Mazurkas by Chopin, and pieces from Vivaldi's *The Four Seasons*. These composers cover three different musical eras, namely the Baroque, Classical, and Romantic periods. Table 3.2 shows a list of the musical pieces underlying the recordings. For each musical piece, our database contains several versions that are performed by different orchestras, conductors, and

Table 3.2: The music collections $\mathbb{D}_1$ used for training and $\mathbb{D}_2$ used for testing. Duration format hh:mm:ss.

	Composer	Piece	Genre	Versions	Dur	⌀ Dur
Training set $\mathbb{D}_1$	Beethoven	Op. 67, Mov. 1	symphony	10	1:12:07	0:07:13
		Op. 67, Mov. 2		10	1:44:53	0:10:29
		Op. 67, Mov. 3		10	1:02:53	0:06:17
		Op. 67, Mov. 4		10	1:48:00	0:10:48
	Chopin	Op. 17, No. 4	mazurka	62	4:29:23	0:04:21
		Op. 24, No. 2	(piano solo)	64	2:26:38	0:02:17
		Op. 30, No. 2		33	0:46:52	0:01:25
		Op. 63, No. 3		86	3:05:06	0:02:09
		Op. 68, No. 3		51	1:25:58	0:01:41
	Vivaldi	RV 315, Mov. 1	violin concerto	7	0:37:40	0:05:23
		RV 315, Mov. 2		7	0:17:23	0:02:29
		RV 315, Mov. 3		7	0:20:40	0:02:57
	Σ			357	19:17:32	
Test set $\mathbb{D}_2$	Beethoven	Op. 55, Mov. 1	symphony	4	1:06:34	0:16:38
		Op. 55, Mov. 2		4	1:02:52	0:15:43
		Op. 55, Mov. 3		4	0:23:41	0:05:55
		Op. 55, Mov. 4		4	0:46:07	0:11:32
	Chopin	Op. 7, No. 1	mazurka	53	2:01:15	0:02:17
		Op. 24, No. 1	(piano solo)	61	2:53:48	0:02:51
		Op. 33, No. 2		67	2:40:38	0:02:24
		Op. 33, No. 3		50	1:29:22	0:01:47
		Op. 68, No. 4		62	2:33:58	0:02:29
	Vivaldi	RV 269, Mov. 1	violin concerto	7	0:24:17	0:03:28
		RV 269, Mov. 2		7	0:20:20	0:02:54
		RV 269, Mov. 3		7	0:30:46	0:04:24
	Σ			330	16:13:37	

soloists. To make our results comparable to prior work, we use audio datasets that are similar to the one used in the study by Grosche and Müller [72]. The datasets comprise recordings of some of Frédéric Chopin's Mazurkas, which have been collected within the Mazurka Project [168].

There are two disjoint sets: $\mathbb{D}_1$ (357 recordings, 62 867 shingles) is used for training the dimensionality reduction methods, and $\mathbb{D}_2$ (330 recordings, 52 332 shingles) is used for evaluating the retrieval quality based on the embedded shingles. Furthermore, circular chroma shifts are applied to the training set, which simulate musical transpositions and increase the number of shingles used for training by a factor of twelve. This process can be seen as a type of data augmentation. Both training and test sets are musically related, as they contain the same composers and the same music genres. However, the musical pieces in both sets are different.

3.5.2 Evaluation Procedure

In our testing stage, we use the dataset $\mathbb{D}_2$, which is independent of the training set $\mathbb{D}_1$. When performing retrieval using a query Q, we compute $\delta_{\mathrm{D}}(\boldsymbol{S}^{\mathrm{Q}})$ for all $\mathrm{D} \in \mathbb{D}_2$ and obtain a ranked list of documents, as described in Section 3.3. We exclude the document containing the query from the database $\mathbb{D}_2$ so that there are no trivial retrieval results.

For evaluating the results, we consider three evaluation measures. First, we use precision at one (P@1), which is 1 if the top-ranked document is relevant, and 0 otherwise. However, not only the top rank is of relevance in our retrieval scenario. This is taken into account by our second evaluation measure, called R-precision (P_R). Here, $R \in \mathbb{N}$ denotes the number of relevant documents for a given query. Note that this number may be different for different queries. For example, $\mathbb{D}_2$ comprises 53 versions of Chopin's Mazurka Op. 7, No. 1. For a query from one of these versions, the number of relevant documents is $R = 52$ (since we exclude the document containing the query from the database). P_R is defined as the proportion of relevant documents among the first R ranks. Third, we use average precision, which is a standard evaluation measure for information retrieval that takes the entire list of ranks into account. It is defined as the mean of the precision scores for the ranks that correspond to relevant documents. This measure is not as well interpretable as P@1 or P_R, but it is the most comprehensive evaluation measure that we use. For a more detailed explanation, we refer to the book by Manning et al. [115, Chapter 8].

For our experiments, we use a set of queries, which we create by equidistantly sampling 10 queries from each recording of our test set $\mathbb{D}_2$, resulting in 3300 queries. The evaluation results are then averaged over all queries. In the case of average precision, the averaged measure is referred to as mean average precision (MAP).

3.5.3 Brute Force

As a baseline, we perform a first retrieval experiment based on the original audio shingles without dimensionality reduction (see Section 3.4.1). Since no training is involved, we report the results in Table 3.3a for both the training set $\mathbb{D}_1$ and the test set $\mathbb{D}_2$. For example, in the case of the test set $\mathbb{D}_2$, we achieved a P@1 value of 0.996, which means that only 13 of the 3300 queries did not yield a relevant document on the top rank. Furthermore, the MAP value of 0.972 indicates that almost all relevant documents appear at the beginning of the ranked list. The results for the training set are similar, indicating a comparable complexity of both datasets.

The parameters and feature design of the shingles are chosen in such a way that the brute-force approach yields close to perfect results for the given task. For a comparison, we also perform an experiment using a temporal alignment procedure, similar to the classical state-of-the-art approaches for cover song retrieval by Serrà et al. [175, 176, 177]. In essence, these approaches are based on the combination of enhanced chroma representations with non-linear temporal alignment procedures. In our experiments, we perform

(a)

BF – Shingles			
Dataset	**P@1**	**P_R**	**MAP**
$\mathbb{D}_1$	0.993	0.936	0.966
$\mathbb{D}_2$	0.996	0.941	0.972

(b)

BF – SDTW, CENS (1 Hz)			
Dataset	**P@1**	**P_R**	**MAP**
$\mathbb{D}_1$	0.991	0.947	0.971
$\mathbb{D}_2$	0.994	0.947	0.975

(c)

BF – SDTW, Chroma (10 Hz)			
Dataset	**P@1**	**P_R**	**MAP**
$\mathbb{D}_1$	0.995	0.966	0.980
$\mathbb{D}_2$	0.999	0.978	0.989

(d)

	GRO [72]			PCA		
K	**P@1**	**P_R**	**MAP**	**P@1**	**P_R**	**MAP**
40	0.930	0.773	0.832	0.991	0.926	0.964
60	0.975	0.871	0.921	0.993	0.938	0.971
80	0.987	0.903	0.948	0.994	0.942	0.973

(e)

	PCA			DNN		
K	**P@1**	**P_R**	**MAP**	**P@1**	**P_R**	**MAP**
3	0.603	0.550	0.580	0.671	0.647	0.683
4	0.807	0.714	0.754	0.772	0.718	0.755
5	0.830	0.712	0.758	0.821	0.766	0.806
6	0.857	0.724	0.771	0.890	0.816	0.856
7	0.888	0.739	0.790	0.908	0.827	0.869
8	0.904	0.754	0.806	0.936	0.856	0.898
9	0.927	0.778	0.834	0.946	0.867	0.910
10	0.945	0.811	0.868	0.954	0.877	0.919
11	0.951	0.813	0.872	0.957	0.888	0.927
12	0.957	0.819	0.877	0.964	0.887	0.928
15	0.974	0.846	0.904	0.969	0.901	0.940
20	0.985	0.878	0.932	0.970	0.905	0.942
30	0.990	0.908	0.952	0.981	0.927	0.959

Table 3.3: Retrieval results **(a)** for the shingle-based brute-force (BF) approach, **(b)** for an alignment-based brute-force approach using CENS features (1 Hz), **(c)** for an alignment-based brute-force approach using chroma features (10 Hz) without CENS post-processing, **(d)** for PCA-based dimensionality reduction of individual chroma vectors (GRO) [72] and entire shingles (PCA, proposed) using the test set $\mathbb{D}_2$, and **(e)** for our proposed dimensionality reduction methods using the test set $\mathbb{D}_2$.

subsequence dynamic time warping (SDTW) [128, Chapter 7] to align the feature sequences of a query and a database document.[9] We use the Euclidean distance, the step size condition $\Sigma := \{(2,1),(1,2),(1,1)\}$, and the weights (2, 1, 1) for vertical, horizontal, and diagonal steps, respectively. As a result of SDTW, we obtain a matching function and the minimum of this matching function is used as a distance measure for ranking the documents. Table 3.3b shows the retrieval results for this experiment, using the CENS features described in Section 3.3. These results are very close to the results of the shingle-based brute-force approach. This confirms that in our music scenario, no alignment procedure is needed when using CENS processing.

One main motivation for using CENS smoothing is to introduce robustness to local tempo variations. When an alignment procedure is used, such smoothing is not needed. Therefore, we also conducted an alignment experiment, using the original chroma features without CENS post-processing. In this setting, the feature rate is 10 Hz instead of 1 Hz. Table 3.3c shows the retrieval results using these features. In the case of the test set $\mathbb{D}_2$, we achieved a P@1 value of 0.999, which means that almost all queries

[9] The approaches of Serrà et al. [175, 176] use local alignment procedures that align subsequences of the query to subsequences of the database document (e.g., Smith–Waterman, or $Q_{\max}$ algorithm). This is motivated by the task of popular music cover song retrieval where the query is a complete recording. In this case, query and relevant database documents typically have a different structure. However, in our case, we are dealing with Western classical music, and the query is only a 20-second excerpt. Therefore, we can expect that the query is entirely represented as a musically corresponding subsequence in the relevant database documents. Under this assumption, subsequence dynamic time warping (SDTW) is more or less equivalent to the Smith–Waterman algorithm.

yield a relevant document on the top rank. In all evaluation measures, we see small improvements over the shingle-based brute-force approach. However, this goes along with a dramatic increase in runtime. The runtime for the overall retrieval experiment in our setting increases from about a minute for the shingle-based brute-force approach (1 Hz features) to several hours for the alignment-based approach using 10 Hz features. We describe further aspects related to runtime in Section 3.5.7.

In summary, we showed that we can get close-to-perfect results with the brute-force approaches. In other words, when only looking at retrieval quality, the problem of cross-version retrieval for Western classical music can be regarded as being largely solved. However, brute-force approaches are time-consuming. The main focus of this chapter is efficiency, and we want to see to which extent we can keep the retrieval quality while reducing the shingle dimensionality. Therefore, in the following, we are not aiming for improving the brute-force approaches, but for keeping a comparable result while using low-dimensional embeddings of the audio shingles. If not mentioned otherwise, brute force always refers to the shingle-based brute-force approach in the following.

3.5.4 PCA

As the second approach, we apply dimensionality reduction with PCA as described in Section 3.4.2. We use the training set $\mathbb{D}_1$ to learn the PCA basis and evaluate the approach with the test set $\mathbb{D}_2$. Table 3.3d shows the evaluation results for two different PCA-based reduction strategies: The left columns (GRO) refer to the reduction of individual chroma vectors as done by Grosche and Müller [72], and the right columns (PCA) refer to the proposed reduction of entire shingles. The rows correspond to the considered dimensionalities 40, 60, and 80. Let us take the case of $K = 40$ as an example. In the first approach (GRO), each chroma vector is reduced to two dimensions, leading to the dimensionality of $K = 2 \cdot 20 = 40$. This strategy resulted in a MAP value of 0.832. In the shingle-based reduction (PCA), an entire 240-dimensional feature sequence is reduced altogether to a 40-dimensional vector. This approach led to a MAP value of 0.964. In general, our experiments show that a shingle-based reduction leads to better retrieval results. This is not surprising, because this approach can exploit temporal redundancies for dimensionality reduction. In the following, we aim to reduce the dimensionality to a degree that would not have been possible with the first approach (GRO). Columns 2–4 of Table 3.3e show the evaluation results obtained with our shingle-based approach for much lower dimensionalities. The retrieval quality consistently increases with an increase of dimensionality from a MAP value of 0.580 for $K = 3$ to 0.952 for $K = 30$. Let us consider the dimensionalities of 6 and 12 as example cases. For $K = 6$, the P@1 value is 0.857, i.e., for 472 of the 3300 queries the top-ranked document was not relevant. For $K = 12$ (P@1: 0.957), this is the case for only 142 queries.

3.5.5 Neural Network with Triplet Loss

As the third approach, we apply dimensionality reduction with a DNN as described in Section 3.4.3. For training the neural network, we used triplets of shingles from the training set $\mathbb{D}_1$. They were generated with the constraint that the central time position of the anchor and positive shingles correspond to the same musical position in different versions of the same piece. The negative shingle does not musically correspond to the anchor shingle. To generate such musically meaningful triplets, we needed to compute musically corresponding time positions in all versions of the same pieces in a pre-processing stage. We used a dynamic-time-warping-based music synchronization approach [128, Chapter 3] for this purpose. Furthermore, random circular shifts along the chroma axis were applied to avoid biasing the network towards the musical keys in our dataset. The shifts used for the anchor and the positive examples were the same, while the shift used for the negative example was chosen independently. This triplet generation procedure leads to a combinatorial explosion of possible triplets. For that reason, not all possible triplets were provided to the network during training. We defined an "epoch" to consist of 2000 mini batches with a mini-batch size of 128 triplets, used for mini-batch gradient descent with the Adam optimizer [89].[10] In our first experiments, we fixed $\alpha = 1.3$ (see Section 3.5.6 for a discussion) as well as a learning rate of 10^{-3} and trained a neural network for 10 epochs. It turned out that a larger number of epochs did not improve the retrieval results.

Columns 5–7 of Table 3.3e show the evaluation results for a range of different dimensionalities from 3 to 30 using the test set $\mathbb{D}_2$. In general, the retrieval quality increases with an increase of K from a MAP value of 0.683 for $K = 3$ to 0.959 for $K = 30$. Let us consider some cases as examples. For $K = 6$, the P@1 value is 0.890, i.e., for 363 of the 3300 queries, the top-ranked document was not relevant. For $K = 12$ (P@1: 0.964), this is the case for only 119 queries. Compared to the PCA-based approach, the neural network especially improves the retrieval results for smaller dimensionalities like 6 or 8, where the MAP value is greater by more than 0.08 (e.g., $K = 8$, MAP for PCA: 0.806 and for DNN: 0.898). We observe rather small improvements in P@1, but there is a considerable increase of P_R (e.g., $K = 8$, P_R for PCA: 0.754 and for DNN: 0.856).[11]

3.5.6 Influence of α

In the following, we analyze the influence of the parameter α, which is used in the loss function as defined in Equation 3.8. This parameter can be interpreted as the margin between $d(\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{p}})$ and $d(\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{n}})$. Figure 3.4 shows the evaluation results (MAP) for various α values. For this experiment, we only use the

[10] Other triplet loss studies sometimes control the triplet generation by a specific procedure called "semi-hard triplet mining" [173]. Preliminary experiments (not reported here) showed that there are no improvements by this method in our case. Therefore, we do not apply this procedure in the experiments reported in this chapter.

[11] We also carried out some informal experiments, where, instead of the triplet loss, we used the contrastive loss [75] and the hinge loss [65, 90]. The results of our informal experiments (not reported here) showed that the triplet and the hinge loss perform similarly. The contrastive loss also performs similarly for low dimensions (below $K = 12$) but worse for higher dimensions.

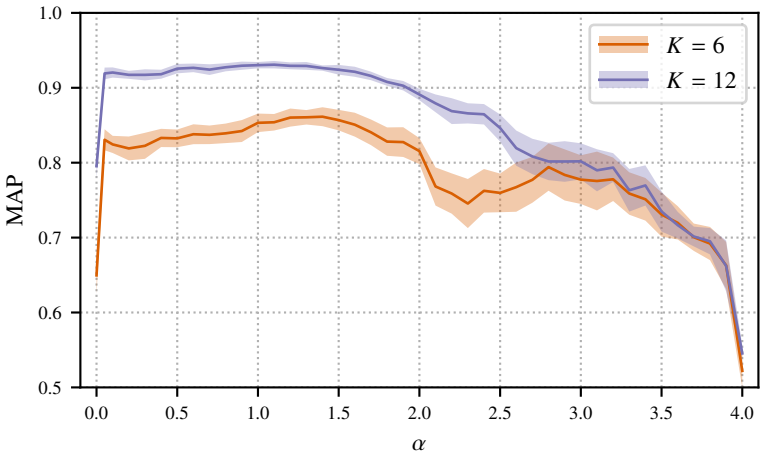


Figure 3.4: MAP evaluation results for dimensionality reduction with various neural networks using the triplet loss with different α values.

dimensionalities of $K = 6$ and $K = 12$ as examples to illustrate general tendencies. For a given α and K, 25 neural networks were trained for 10 epochs with different random initializations. Then, they were used for dimensionality reduction in the retrieval scenario, resulting in 25 MAP values. From these, we computed the mean (μ) and the standard deviation (σ). The solid lines show μ, and the light areas show $\pm\sigma$ around μ. For both $K = 6$ and $K = 12$, we see similar trends: $\alpha = 0$ achieves rather bad results. In this case, no margin is enforced, and the loss is zero as soon as $d(\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{p}}) \leq d(\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{n}})$. However, increasing α only slightly leads to a clear improvement in MAP. Any α in the range of $[0.1, 1.7]$ produces results of similar quality. For $\alpha > 1.7$, the results strongly decrease, which can be explained as follows: Only a small positive margin is needed for retrieving the correct versions. Using a large margin brings no benefit for retrieval but makes the training much harder. In summary, for $0.1 \leq \alpha \leq 1.7$, we see a stable overall behavior of the results with a small standard deviation, showing robustness to the initialization used.

3.5.7 Runtime Experiment

We showed that we can substantially reduce the dimensionality of the audio shingles while keeping their discriminative power. Such low-dimensional embeddings are beneficial when using indexing techniques for efficient nearest neighbor retrieval. To show this property, we conducted an experiment where we computed the distances of the 3300 queries to the documents of our test set $\mathbb{D}_2$ (as done for the previous experiments). We measured the runtimes for the alignment-based approaches (described in Section 3.5.3) and shingle-based approaches. In the case of the shingle-based retrieval, we employed three different nearest neighbor search strategies. The first search strategy is a full search by just computing all distances between the shingles of the database documents and the query shingles. For the second and third search strategies, we used K-d trees, which are standard data structures for searching in multidimensional spaces [18]. In the second strategy (Doc-Trees), we built one tree for each of the 330 documents of the test set and searched for the nearest neighbor to the query in each tree. As a consequence, each document occurred precisely once in the ranked list (as done for the previous experiments). In the third strategy (Db-Tree), we built a single tree for all database documents and searched for the 330 nearest embeddings

(a)

Approach	Full Search	Index (Db-Tree)
BF – SDTW, Chroma (10 Hz)	23 190.1	-
BF – SDTW, CENS (1 Hz)	351.5	-
BF – Shingles ($K = 240$)	23.0	76.9
DNN ($K = 30$)	3.3	5.0
DNN ($K = 12$)	1.8	1.1
DNN ($K = 6$)	1.2	0.4

(b)

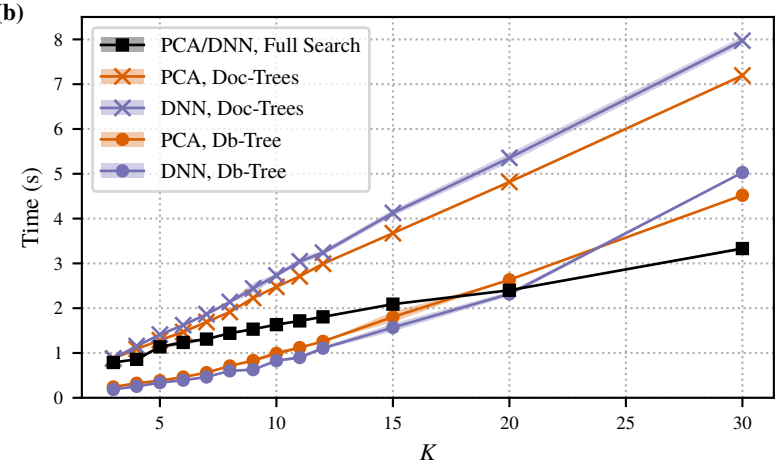


Figure 3.5: Time (in seconds) needed for searching 3300 queries. **(a)** Table with selected approaches including the brute-force approaches. **(b)** Figure showing the runtimes for the shingle-based approaches depending on the embedding dimensionality K, with various nearest neighbor search strategies.

to the query. With this strategy, we were not able to rank all documents of the database because some of the returned embeddings originated from the same document. Note that the reported runtimes depend on the used implementations.[12] So, rather than focusing on the absolute times, we want to emphasize the relative tendencies and the orders of magnitude.

Figure 3.5a presents the runtimes for selected settings, averaged over several iterations of the retrieval experiment. The first column specifies the retrieval approach, the second column lists the runtimes for the full search strategy, and the third column lists the runtimes for the Db-Tree search strategy. For the alignment-based approach using 10 Hz features (first row), the runtime is about 6.5 hours. When using 1 Hz features (second row), the runtime decreases to about 6 minutes. It is not surprising that the first alignment-based approach is much slower since the feature rate is ten times higher, and the alignment algorithms are of quadratic complexity. For the brute-force shingle approach ($K = 240$, third row), the runtime is significantly lower than for both alignment-based approaches. It takes 23.0 seconds for the full search strategy and 76.9 seconds for the Db-Tree search strategy. In our setting and with the used implementations, the Db-Tree strategy is slower than the full search for $K = 240$. It is well known that K-d trees degenerate for high dimensions [116]. With dimensionality reduction to $K = 30$ or $K = 12$ (fourth

[12] We performed our experiments using Python 3.6.5 on a computer with an Intel Xeon CPU E5-2620 v4 (2.10 GHz) and 31 GiB RAM. For the alignment-based approaches, we use the SDTW implementation of librosa 0.7.1 [119], which is written in Python and accelerated by the just-in-time compiler numba. For the full search, we use the efficient pairwise-distance calculation of scipy 1.0.1 [199], which calls a highly optimized implementation in C. For the K-d trees (using a default leaf size of 30), we use the implementation of scikit-learn 0.20.1 [142], which is written in Cython.

and fifth row), both search strategies are in a similar range (e.g., for $K = 12$, full search: 1.8 seconds, Db-Tree: 1.1 seconds). For lower dimensions ($K = 6$, sixth row), the Db-Tree search strategy substantially accelerates the nearest neighbor search (full search: 1.2 seconds, Db-Tree: 0.4 seconds).

Figure 3.5b shows the runtime (μ and $\pm\sigma$ for 100 iterations of the retrieval experiment) for the dimensionality reduction approaches. For the full search strategy, we see an almost linear relationship between the dimensionality K and the runtime. This strategy is independent of the underlying data distribution. For that reason, we do not distinguish between PCA- and DNN-based dimensionality reduction for the full search strategy. For the tree-based search strategies (Db-Tree and Doc-Trees), however, the data distributions of the PCA- and DNN-based embeddings lead to different search times. We also see an almost linear relationship for the Doc-Trees search strategy for both the PCA- and DNN-based retrieval approaches. In general, for this strategy, the runtime is higher than for the full search. In our case, the data size of a single document is too small for the Doc-Trees strategy to give any benefit over the full search strategy. For the Db-Tree strategy, we see a growth of runtime with growing K that is slightly stronger than linear. When the dimensionality falls below 15, the Db-Tree strategy begins to provide benefits for a fast nearest neighbor search.

We want to emphasize again that the absolute runtimes are implementation-dependent. Therefore, we instead want to highlight some general tendencies: The shingle-based approaches are significantly faster than the alignment-based approaches. In general, the experiments confirm that lower dimensionalities accelerate the nearest neighbor search. In particular, when using small dimensions (below 15 in our setting), we can further speed up the search by standard multidimensional indexing strategies. In Chapter 4, we will explore indexing approaches for our cross-version retrieval task in more detail.

3.6 Extended Experiments

In this section, we investigate the scalability and generalizability of our approach by evaluating the embedding methods on a larger and more diverse dataset. In this way, we also provide deeper insights into the benefits and limitations of our dimensionality reduction approaches. First, in Section 3.6.1, we describe our extended dataset, which is only used for testing. Then, in Section 3.6.2, we discuss the evaluation results obtained using the embedding methods trained on the smaller training set described in Section 3.5.1. Next, in Section 3.6.3, we investigate the discriminatory capacity of the low-dimensional embeddings by using a longer query length employing multiple shingles per query. Finally, in Section 3.6.4, we analyze the distances that appear in the nearest neighbor search to better understand the complexity of the retrieval problem depending on the composers and genres of classical music.

Composer	Piece	Genre	Versions	Dur	⌀ Dur
Test set $\mathbb{D}_2$, consisting of 12 different pieces (see Table 3.2)			330	16:13:37	
Brahms	Op. 83, Mov. 1	piano concerto	6	1:44:23	0:17:24
	Op. 83, Mov. 2		6	0:52:08	0:08:41
	Op. 83, Mov. 3		6	1:12:31	0:12:05
	Op. 83, Mov. 4		6	0:55:08	0:09:11
Mahler	Symphony No. 1, Mov. 1	symphony	5	1:15:16	0:15:03
	Symphony No. 1, Mov. 2		5	0:35:40	0:07:08
	Symphony No. 1, Mov. 3		5	0:54:42	0:10:56
	Symphony No. 1, Mov. 4		5	1:36:53	0:19:23
Mozart	KV 550, Mov. 1	symphony	5	0:35:40	0:07:08
	KV 550, Mov. 2		5	0:50:12	0:10:02
	KV 550, Mov. 3		5	0:20:10	0:04:02
	KV 550, Mov. 4		5	0:32:54	0:06:35
Mussorgsky	Pict. at an Exhib., Promenade	character piece (piano solo or orchestral arrangement)	6	0:09:07	0:01:31
	Pict. at an Exhib., No. 1		6	0:14:50	0:02:28
	Pict. at an Exhib., No. 2		6	0:24:48	0:04:08
	Pict. at an Exhib., No. 3		6	0:06:49	0:01:08
	Pict. at an Exhib., No. 4		6	0:17:51	0:02:58
	Pict. at an Exhib., No. 5		6	0:07:39	0:01:16
	Pict. at an Exhib., No. 6		6	0:13:51	0:02:18
	Pict. at an Exhib., No. 7		6	0:08:13	0:01:22
	Pict. at an Exhib., No. 8		6	0:11:31	0:01:55
	Pict. at an Exhib., No. 9		6	0:20:18	0:03:23
	Pict. at an Exhib., No. 10		6	0:32:41	0:05:27
Schumann	Op. 54, Mov. 1	piano concerto	5	1:13:59	0:14:48
	Op. 54, Mov. 2		5	0:24:43	0:04:57
	Op. 54, Mov. 3		5	0:51:47	0:10:21
Tchaikovsky	Op. 23, Mov. 1	piano concerto	8	2:33:39	0:19:12
	Op. 23, Mov. 2		8	0:50:34	0:06:19
	Op. 23, Mov. 3		8	0:52:59	0:06:37
	Op. 35, Mov. 1	violin concerto	6	1:47:27	0:17:54
	Op. 35, Mov. 2		6	0:38:30	0:06:25
	Op. 35, Mov. 3		6	0:54:06	0:09:01
Wagner	WWV 86 B, Act 1	opera	18	19:15:20	1:04:11
Σ			535	59:49:56	

Table 3.4: Extended dataset $\mathbb{D}_3$. This dataset includes the test set $\mathbb{D}_2$. Duration format hh:mm:ss.

3.6.1 Extended Dataset

To test the scalability and generalizability of our approach, we compiled an extended dataset $\mathbb{D}_3$, which is listed in Table 3.4. Including the test set $\mathbb{D}_2$ (see Table 3.2), the extended dataset additionally comprises a variety of further composers and genres, such as piano and violin concertos (Brahms, Schumann, Tchaikovsky), symphonies (Mahler, Mozart), opera music (Wagner), as well as character pieces in piano solo and orchestral versions (Mussorgsky). The dataset $\mathbb{D}_3$ consists of 535 recordings (205 522 shingles) and contains about 60 hours of audio material, compared to 16 hours of the previous test set $\mathbb{D}_2$.

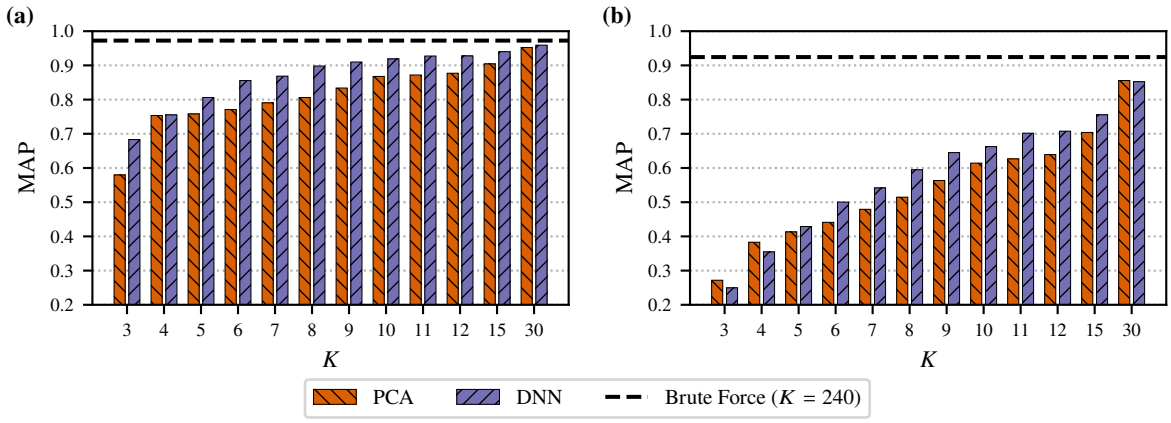


Figure 3.6: Evaluation results for **(a)** the smaller test set $\mathbb{D}_2$ and **(b)** the extended dataset $\mathbb{D}_3$.

3.6.2 Evaluation

For our extended experiments, we keep the settings from the previous experiments and apply the same embedding methods as described in Section 3.4. In particular, the embedding methods were trained on the smaller training set $\mathbb{D}_1$ (see Table 3.2) as before, and are now evaluated with the extended dataset $\mathbb{D}_3$, containing composers and genres of classical music that are not included in the training set. The retrieval for a larger and more diverse dataset obviously constitutes a more challenging task. For this reason, we can expect the retrieval results to decrease.

Figure 3.6 shows the MAP evaluation results for different embedding dimensionalities K for the smaller test set $\mathbb{D}_2$, as reported in the previous Section 3.5 (left) and for the extended dataset $\mathbb{D}_3$ (right). As expected, we see a decrease in retrieval quality for the larger dataset. The results for the brute-force approach decrease from a MAP of 0.996 for $\mathbb{D}_2$ to 0.924 for $\mathbb{D}_3$. The results based on the embedding approaches decrease even more. In particular, the smaller dimensionalities result in a MAP of less than 0.6, e.g., for $K = 6$, the MAP is 0.441 and 0.500 for PCA and DNN, respectively. These findings confirm our assumption that the extended dataset constitutes a harder task and leads to an increased probability for false negatives. One reason for the task's increased difficulty is that there is a higher potential for confusion between the documents due to the increased dataset size. Another reason is the increased diversity of database documents.

To gain more insights into these results, we now analyze the evaluation for each of the individual composers of the dataset. In the following, we fix the dimensionality of $K = 12$ as a typical example that gives a good trade-off between retrieval quality and speed, as shown in the previous experiments. Table 3.5 shows the detailed evaluation results for $K = 12$. The rows correspond to the composers of the dataset, and the last row reports the averaged evaluation measures over all queries and all composers. Note that composers with many queries have a stronger impact on this average than composers with few queries. The queries from the Chopin recordings result in a MAP value of 0.932 with the brute-force approach, 0.686 with PCA, and 0.825 with DNN. These numbers are lower than those obtained for the smaller test set (0.972 for

	Queries	Brute Force			PCA ($K = 12$)			DNN ($K = 12$)		
		P@1	P_R	MAP	P@1	P_R	MAP	P@1	P_R	MAP
Beethoven	160	0.963	0.827	0.879	0.631	0.456	0.521	0.669	0.477	0.538
Chopin	2930	0.996	0.876	0.932	0.899	0.626	0.686	0.936	0.765	0.825
Vivaldi	210	0.967	0.921	0.953	0.790	0.645	0.716	0.819	0.646	0.728
Brahms	240	0.992	0.960	0.976	0.754	0.543	0.619	0.796	0.593	0.667
Mahler	200	0.950	0.920	0.940	0.845	0.682	0.755	0.700	0.539	0.617
Mozart	200	1.000	0.864	0.907	0.630	0.390	0.443	0.560	0.349	0.399
Mussorgsky	660	0.967	0.834	0.877	0.718	0.482	0.545	0.642	0.452	0.516
Schumann	150	0.980	0.893	0.918	0.553	0.400	0.462	0.653	0.472	0.553
Tchaikovsky	420	0.983	0.892	0.925	0.729	0.493	0.566	0.621	0.492	0.543
Wagner	180	0.983	0.849	0.912	0.761	0.625	0.676	0.722	0.584	0.636
⌀ over queries		0.987	0.877	0.924	0.818	0.577	0.639	0.818	0.646	0.708

Table 3.5: Composer-wise evaluation results using the extended dataset $\mathbb{D}_3$.

brute force, 0.877 for PCA, 0.928 for DNN, see Table 3.3). Since the Chopin recordings of $\mathbb{D}_2$ and $\mathbb{D}_3$ are identical, we can conclude that the lower evaluation measures are due to the larger dataset size of $\mathbb{D}_3$. In particular, PCA suffers from using the larger test dataset. The moderate loss for the DNN approach could point towards overfitting because Chopin is also the most prominent composer of the training set. The queries from the Mahler recordings result in a MAP value of 0.940 with the brute-force approach, 0.755 with PCA, and 0.617 with DNN. Here, the DNN is considerably worse than the linear PCA-based projection. The late-romantic style of Mahler could be too different from the styles of the training set. The queries from the Schumann recordings result in a MAP value of 0.918 with the brute-force approach, 0.462 with PCA, and 0.553 with DNN. In this case, the DNN achieves better results than those of the PCA. Note that Schumann is not part of the training set and that the respective work is a piano concerto, which is a classical music genre that is also not covered in the training set.

A possible explanation for the poorer results of the embedding methods could be that the embeddings are just overfitted to the composers and styles of the training set and are not very discriminative in general. In the next section, we will question this argument by considering longer query audio fragments.

3.6.3 Dependency on Query Length

A possible explanation for the results of the previous section is that the query length of 20 seconds is not discriminative enough to identify all versions of the same piece. To analyze this hypothesis, we increase the query length in the following experiments. An obvious way to do this is to increase the query shingle length. However, we want to keep our shingle size fixed to keep the same database structure for different query lengths. Therefore, instead of increasing the query shingle length, we use multiple successive shingles for each query.

Recall that, in our previous approach (see Section 3.3), we compare a query (Q) and a document (D) by performing a nearest neighbor search of a single query shingle $\boldsymbol{S}^{\mathrm{Q}}$ and all shingles from the set $\mathbb{S}^{\mathrm{D}}$

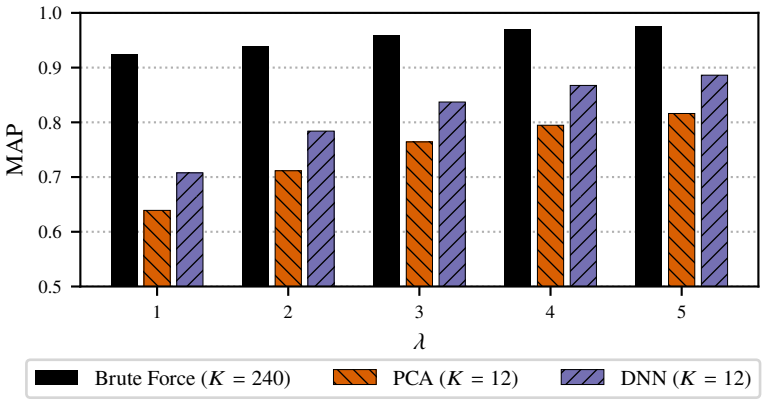


Figure 3.7: Evaluation results with varying λ for the extended dataset $\mathbb{D}_3$.

of document submatrices (see Equation 3.2). The squared Euclidean distance to the nearest neighbor $\delta_{\mathrm{D}}(\boldsymbol{S}^{\mathrm{Q}}) \in \mathbb{R}_{\geq 0}$ is regarded as the document-wise distance between Q and D and used to rank all the database documents. Now, instead of a single query shingle $\boldsymbol{S}^{\mathrm{Q}}$, we collect a set of successive shingles $\mathbb{S}^{\mathrm{Q}}$ from the query recording, as done for the documents (see Equation 3.2). Instead of using a hop size $H = 1$ (as for the documents), we use a hop size of $H = L/2 = 10$. Denoting the number of shingles per query by $\lambda := |\mathbb{S}^{\mathrm{Q}}|$, a query covers $(\lambda - 1) \cdot 10 + 20$ seconds of audio content. For each of these shingles, we compute the document-wise distance δ_{D} as previously (see Equation 3.6). The resulting shingle-wise distances are simply averaged for obtaining a single distance value between the query and the database document:

$$\overline{\delta_{\mathrm{D}}}(\mathbb{S}^{\mathrm{Q}}) = \frac{1}{\lambda} \sum_{\boldsymbol{S} \in \mathbb{S}^{\mathrm{Q}}} \delta_{\mathrm{D}}(\boldsymbol{S}). \tag{3.10}$$

Finally, all database documents $\mathrm{D} \in \mathbb{D}$ are ranked by their averaged distances $\overline{\delta_{\mathrm{D}}}(\mathbb{S}^{\mathrm{Q}}) \in \mathbb{R}_{\geq 0}$ in ascending order. Note that our previous experiments are the special case of $\lambda = 1$ (query length: 20 seconds).

For our experiments, we sample 10 queries from each recording in an equidistant way, as before. Note that each query now consists of multiple shingles. Figure 3.7 shows the MAP evaluation results for $K = 12$ using different query lengths for the extended dataset. The retrieval quality considerably improves with increasing query length, e.g., the brute-force approach improves from a MAP value of 0.924 for $\lambda = 1$ to 0.976 for $\lambda = 5$. Similarly, the MAP values for the PCA- and DNN-based approaches increase strongly with the query length.

Table 3.6 shows the results for each of the composers of the dataset for $\lambda = 5$ (query length: 60 seconds). For Chopin, the DNN (MAP: 0.937) outperforms PCA (MAP: 0.813) and comes close to brute force (MAP: 0.974). For Mahler, PCA (MAP: 0.935) is slightly better than the DNN (MAP: 0.891). For Schumann's piano concerto, the DNN (MAP: 0.840) substantially outperforms PCA (MAP: 0.681). In contrast to the Chopin results, this result cannot be explained by overfitting because neither Schumann nor any piano concerto is part of the training set. Furthermore, we achieved a MAP value of 0.967 for the pieces by Brahms using the DNN approach. This result is the best among all composers for this approach,

	Queries	Brute Force			PCA ($K = 12$)			DNN ($K = 12$)		
		P@1	P_R	MAP	P@1	P_R	MAP	P@1	P_R	MAP
Beethoven	160	0.956	0.921	0.941	0.800	0.598	0.666	0.838	0.633	0.698
Chopin	2930	0.997	0.943	0.974	0.973	0.727	0.813	0.979	0.886	0.937
Vivaldi	210	1.000	0.986	0.996	0.971	0.864	0.918	0.976	0.864	0.925
Brahms	240	1.000	0.998	0.999	0.996	0.912	0.950	0.992	0.943	0.967
Mahler	200	0.985	0.976	0.982	0.960	0.900	0.935	0.935	0.846	0.891
Mozart	200	1.000	0.966	0.980	0.935	0.639	0.704	0.880	0.600	0.670
Mussorgsky	660	0.992	0.937	0.960	0.933	0.703	0.773	0.865	0.697	0.756
Schumann	150	0.993	0.977	0.983	0.800	0.630	0.681	0.920	0.780	0.840
Tchaikovsky	420	0.998	0.984	0.990	0.952	0.776	0.839	0.929	0.788	0.838
Wagner	180	1.000	0.956	0.975	0.978	0.884	0.922	0.989	0.904	0.940
⌀ over queries		0.995	0.953	0.976	0.956	0.744	0.816	0.951	0.835	0.886

Table 3.6: Composer-wise evaluation results using the extended dataset $\mathbb{D}_3$, for $\lambda = 5$ (query length: 60 seconds).

even better than for the composers of the training set. These findings show a certain generalizability of the DNN embedding method.

In summary, our experiments show that low-dimensional embeddings need a longer query length to be discriminative enough when using a larger dataset. Note that a query length of 60 seconds is still a medium duration compared to studies for related tasks. For example, in popular music cover song retrieval, an entire recording is often used as a query, e.g., in the approach by Serrà et al. [175] or the work by Casey et al. [39] (using entire recordings as queries, though with a short shingle length of 3 seconds). Furthermore, we showed that, in general, composers outside the training set do not necessarily lead to worse retrieval results, compared to composers contained in the training set. Thus, we can assume a certain generalizability of our approach within the common practice period of Western classical music.

3.6.4 Distance Analysis

In the previous section, we addressed issues of scalability and generalizability in relation to the query length. Now we want to gain further insights into the challenges that occur in the retrieval scenario. For example, beyond rank-based evaluation measures, we want to find out whether particular compositions are easier or harder for retrieval than others, e.g., due to harmonic or melodic characteristics. Recall from Section 3.3 that the ranks are computed on the basis of the distances that appear between queries and documents. The retrieval problem is well behaved if the distances between queries and documents of the same piece of music (relevant documents) are substantially smaller than the distances between queries and documents of different pieces (non-relevant documents). To understand how well behaved our problem is, we now analyze the distances that appear in the nearest neighbor search.

Recall that we have $R \in \mathbb{N}$ relevant documents for a given query. We want to compare the distances to the relevant documents with the distances to the non-relevant documents. Since most of the non-relevant documents should be easily distinguishable from the relevant ones, we restrict our analysis to the most

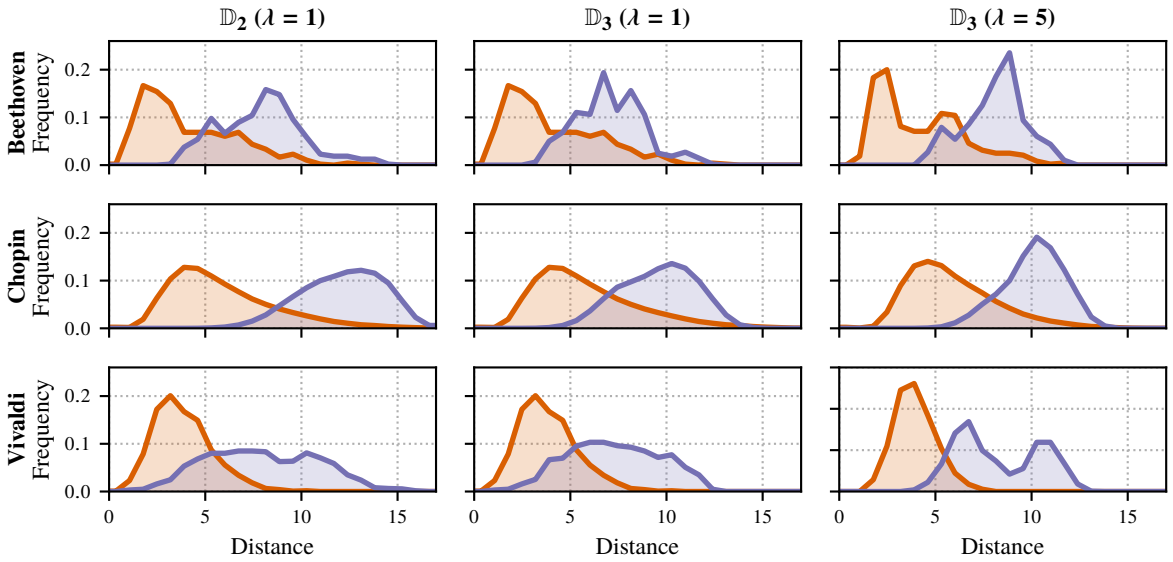


Figure 3.8: Distance distributions for three composers (Beethoven, Chopin, Vivaldi) for the test set $\mathbb{D}_2$ with $\lambda = 1$ (left column), the extended test set $\mathbb{D}_3$ with $\lambda = 1$ (middle column), and the extended test set $\mathbb{D}_3$ with $\lambda = 5$ (right column). All distributions are computed for the brute-force approach ($K = 240$). The orange color refers to distances for relevant documents and the blue color refers to distances for non-relevant documents.

difficult non-relevant documents for the task. To balance the numbers of relevant and non-relevant documents, we only consider the R non-relevant documents with the smallest distances to the given query. Small distances mean that these non-relevant documents have the greatest confusion potential with the relevant documents. In the following, we analyze the relation of the distributions of distances for relevant documents and non-relevant documents, respectively. The relation between these distributions indicates how difficult the retrieval problem is. However, the analysis of the relation between the distributions has to be taken with care because versions with different difficulties are included in a single distribution (where some queries may generally lead to lower distances for positive and negative matches and some queries may generally lead to higher distances). For that reason, a strong separation of the distributions is a sufficient condition for perfect retrieval results, but not a necessary condition. In other words, even if the overlap between the distributions is large, the retrieval may still yield excellent results. In this sense, we regard the distributions only as weak indicators and only evaluate them by visual inspection in the following. For a more statistically rigorous analysis of such distributions, we refer to the study by Casey et al. [39].

Figure 3.8 shows such distributions for the brute-force approach, where the distributions of relevant document distances appear in orange color, and the distributions of non-relevant document distances appear in blue color. The three rows show distributions for the composers that are part of both $\mathbb{D}_2$ and $\mathbb{D}_3$ (Beethoven, Chopin, Vivaldi), and the three columns correspond to different evaluation settings. The left column refers to the smaller test set $\mathbb{D}_2$ with $\lambda = 1$ (query length: 20 seconds), the middle column refers to the extended test set $\mathbb{D}_3$ with $\lambda = 1$ (query length: 20 seconds), and the right column refers to the extended test set $\mathbb{D}_3$ with $\lambda = 5$ (query length: 60 seconds). Considering Chopin in the left column,

we see that the center of the orange distribution is much further to the left than the blue distribution. This means that the distances to the relevant documents are generally smaller than the distances to the non-relevant documents. We also see a little overlap between both distributions, which is caused by the fact that some distances to non-relevant documents are smaller than the greatest distances to relevant documents. However, since the distances that lead to the overlap could relate to different queries, we cannot conclude that this necessarily leads to confusion in the retrieval step. In general, since the overlap is small, the distributions indicate that the retrieval problem is well behaved for Chopin. For Beethoven and Vivaldi (left column), there is more overlap. This suggests that the Chopin pieces are "easier" in the sense that they are more discriminative than other pieces. For the extended dataset (middle column), we see similar tendencies for all three composers. The blue distributions come a bit closer to the orange ones compared to their positions for the smaller dataset (left column). The orange distributions are identical to the ones for the smaller dataset because the distances to the relevant documents are the same. Only the distances to non-relevant documents decrease because the extended dataset contains more non-relevant documents with possibly smaller distances. When using a longer query length ($\lambda = 5$, right column), the orange and blue distributions are better separated for all three composers. The distances for the relevant documents only slightly increase because of the longer query length. Still, the distances to the non-relevant documents increase strongly, which leads to a better separation between the distributions.

So far, we analyzed distributions for the brute-force approach to gain insights into the musical complexity of the retrieval task. In the following, we want to understand the effect of the embedding on the distributions. Figure 3.9 shows further distributions for the extended test set $\mathbb{D}_3$ and five selected composers (Beethoven, Chopin, Vivaldi, Mahler, and Schumann). The left column refers to the brute-force approach ($K = 240$) with $\lambda = 1$ (query length: 20 seconds), the middle column refers to the DNN embedding approach ($K = 12$) with $\lambda = 1$ (query length: 20 seconds), and the right column refers to the DNN embedding approach ($K = 12$) with $\lambda = 5$ (query length: 60 seconds). Note that the absolute distances between the different approaches are not comparable[13], but the relations between orange and blue distributions are meaningful. For the brute-force approach (left column), the new composers of the extended dataset (Mahler, Schumann) behave similarly to the previous ones. The middle column reflects the weaker results for the low-dimensional embeddings with a shorter query length. As expected, there are strong overlaps between the orange and blue distributions. We can also see that there are some distances close to zero in the orange distributions. This can be explained by the neural network training, where the anchor and positive embeddings are pushed close together. However, the anchor and negative embeddings do not seem to be pulled apart the same way. For Chopin, the most prominent composer of the training set $\mathbb{D}_1$, the separation between the distributions is clearer. When using longer queries ($\lambda = 5$, right column), all orange and blue distributions are better separated. This holds for Chopin, where the initial situation ($\lambda = 1$) is better, and for composers with heavily overlapping distributions for $\lambda = 1$, e.g., Schumann.

[13] The distance measure is always the squared Euclidean distance, cf. Equation 3.5. However, for the brute-force approach, we have sequences of 20 non-negative ℓ^2-normalized vectors of size 12, which are then flattened (distance range $[0, 40]$). For the DNN approach, we have real-valued ℓ^2-normalized vectors of size 12 (distance range $[0, 4]$).

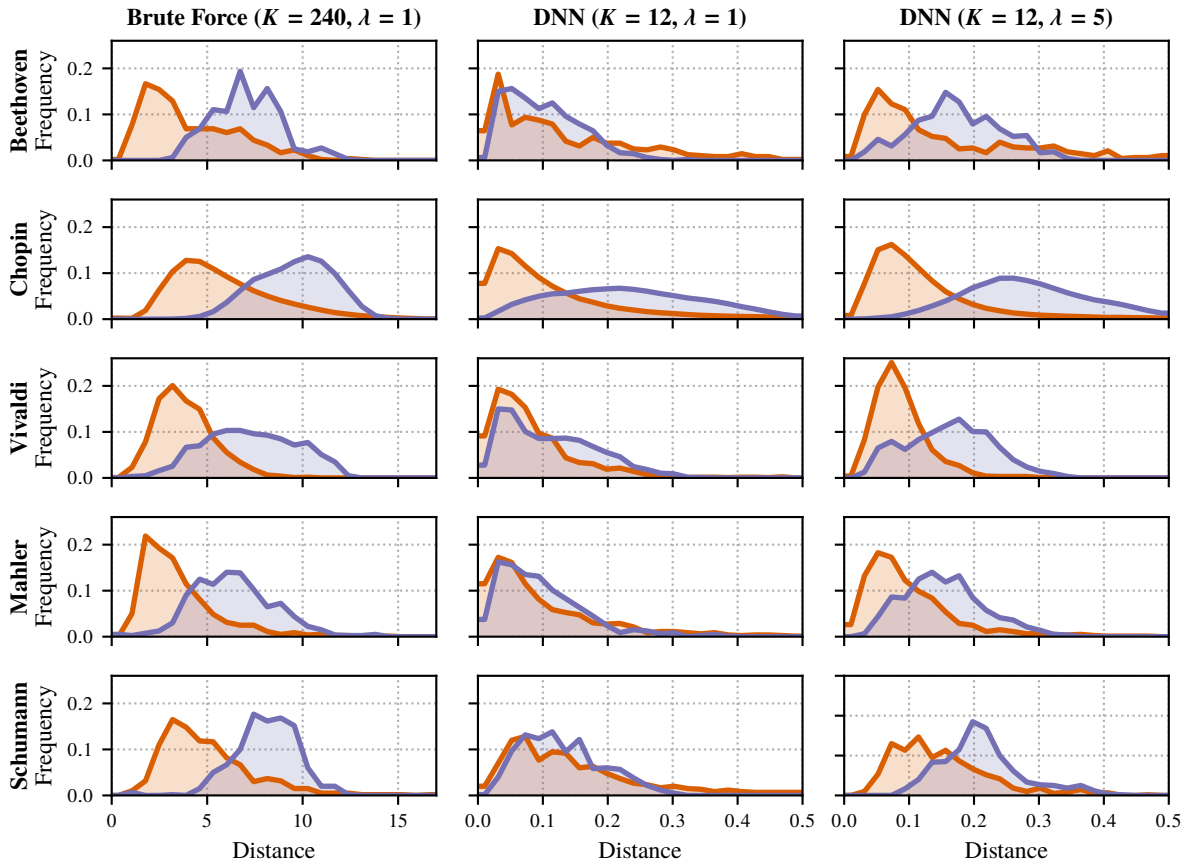


Figure 3.9: Distance distributions for five composers (Beethoven, Chopin, Vivaldi, Mahler, Schumann) for the brute-force approach ($K = 240$) with $\lambda = 1$ (left column), the DNN approach ($K = 12$) with $\lambda = 1$ (middle column), and the DNN approach ($K = 12$) with $\lambda = 5$ (right column). All distributions are computed for the extended test set $\mathbb{D}_3$. The orange color refers to distances for relevant documents and the blue color refers to distances for non-relevant documents.

3.7 Conclusions

In this chapter, we proposed two dimensionality reduction methods for learning compact embeddings of audio shingles (short sequences of chroma vectors) for a cross-version retrieval scenario in the context of Western classical music. We showed that our strategy of reducing entire shingles results in better retrieval quality and faster speed than the previous approach of reducing individual chroma vectors [72]. In our experiments, we strongly reduced the shingles' dimensionality from 240 to below 12, with only a little loss in retrieval quality. Both PCA and neural networks with triplet loss turned out to be effective for this task. In particular, we found that neural networks are beneficial for small dimensionalities between 6 and 12. Such small dimensions allow for indexing by simple nearest neighbor trees, which could be the foundation of fast content-based audio retrieval in large classical music databases where efficiency is an important issue. We also showed that the database size substantially impacts the retrieval results, especially when using dimensionality reduction methods. Applying such techniques, one needs a longer query length to be discriminative enough on a larger dataset. Increasing the query length from 20 to 60

seconds results in a high retrieval accuracy, even for low-dimensional embeddings. We also showed that our approaches generalize to composers of the common practice period of Western classical music that are not contained in the training set. Furthermore, we analyzed the distances that appear in the nearest neighbor search to gain insights into the challenges of the retrieval scenario. For example, we showed that the distance distributions for different composers can differ strongly and indicate that certain composers or pieces are more difficult for retrieval than others. In future work, one may investigate whether training on larger datasets can make the embeddings more robust for shorter query lengths. Furthermore, up to now, we used CENS features as input, which are state-of-the-art for the given task. It would be interesting to investigate if useful embeddings can be learned from less specialized input features, such as "raw" chroma features, spectrograms, or even waveforms. In the next chapter, we will explore a more advanced indexing approach for our cross-version retrieval scenario to further improve the retrieval efficiency.

4 Efficient Cross-Version Music Retrieval Using Graph-Based Index Structures

This chapter is based on [223]. The first author Frank Zalkow is the main contributor to this article. He wrote the paper in collaboration with his supervisor Meinard Müller. Some ideas, developed further in the article, emerged from Julian Brandner's Master thesis [28], supervised by Frank Zalkow and Meinard Müller. Based on the work for this thesis, Frank Zalkow implemented the approaches and conducted the experiments. The Carus publisher provided the larger dataset ($\mathbb{D}_{\text{lg}}$) used in the experiments. We would like to especially thank Johannes Graulich, Ester Petri, and Iris Pfeiffer for the provision.

Flexible retrieval systems are required for conveniently browsing through large music collections. In a particular content-based music retrieval scenario, the user may provide a query audio snippet, and the retrieval system returns music recordings from the collection that are similar to the query. In this scenario, a fast response from the system is essential for a positive user experience. For realizing low response times, one requires index structures that facilitate efficient search operations. One such index structure is the K-d tree, which has already been used in music retrieval systems (see also the previous chapter). As an alternative, we propose to use a modern graph-based index, denoted as *hierarchical navigable small world* (HNSW) graph. As our main contribution, we explore its potential in the context of a cross-version music retrieval application. In particular, we report on systematic experiments comparing graph- and tree-based index structures in terms of the retrieval quality, disk space requirements, and runtimes. Despite the fact that the HNSW index provides only an approximate solution to the nearest neighbor search problem, we demonstrate that it has almost no negative impact on the retrieval quality in our application. As our main result, we show that the HNSW-based retrieval is several orders of magnitude faster. Furthermore, the graph structure also works well with high-dimensional index items, unlike the tree-based structure. Given these merits, we highlight the practical relevance of the HNSW graph for MIR applications.

4.1 Introduction

Ongoing digitization efforts lead to increasingly large music collections. With growing dataset sizes, it can become challenging to find relevant audio documents in such a collection. A paradigm for searching in music databases is known as query-by-example, where the user provides an audio query, and the task is to find audio recordings from the database containing parts or aspects similar to the query [40, 73, 194].

An example of a public query-by-example music retrieval service is the audio fingerprinting application Shazam [202, 203], where the user supplies a query audio snippet, which is then identified by comparing the snippet's fingerprint with fingerprints from a reference database. One reason for Shazam's popularity is the ability to provide music identification results close to instantly. This short response time is possible because of the strict notion of similarity (identity of recordings) between the query and relevant database recordings in the audio identification task, combined with clever indexing techniques [38, 40, 73]. In our study, we address a cross-version retrieval scenario, where we aim to find all performances or versions of a given piece of music, which is specified by a query audio snippet. In related tasks, the user may provide the query by singing or humming a melody [66, 166]. In such cross-version scenarios, the notion of similarity is less strict (identity of the musical piece underlying different recordings), which leads to higher response times compared to fingerprinting services. In the previous chapter, we showed that the runtime of the search procedure in a cross-version retrieval scenario is in the order of a few seconds, even for a small database of about 16 hours. With growing database sizes, this runtime also increases and becomes prohibitive for usage outside academia. In a query-by-example setting, it makes a dramatic difference whether the user has to wait a fraction of a second or a couple of seconds for the results after specifying the query. Efficiency is an important aspect of usability and a critical requirement of information retrieval systems for being practically relevant [62]. In this chapter, we show how to increase the efficiency for our cross-version retrieval scenario by using a modern indexing approach.

Indexing procedures increase the speed of search operations in a database, using specialized data structures, such as inverted file indices [99]. In our context, the nearest neighbor search is essential, where one aims to find the closest item in a database to a given query item. We can classify nearest neighbor search procedures into exact and approximate search approaches. Exact nearest neighbor search procedures (such as K-d trees [18, 61]) guarantee to find the item in the database that is closest to the query. Other approaches relax this requirement. Instead of finding exact nearest neighbors, they only aim to find sufficiently nearby neighbors. In general, this is referred to as approximate nearest neighbor search. A well-known approach of this category is, e.g., LSH [183]. Beyond the distinction of approximate and exact solutions, nearest neighbor search procedures can be categorized according to their algorithmic approach [106, 126]. The main catagories of this distinction are hashing-based, partition-based, and graph-based search approaches. An example of hashing-based procedures is LSH [183], which has already been used in music retrieval studies [72, 123, 164, 170]. Grosche und Müller [72] found that LSH can increase the retrieval efficiency compared to an exhaustive search. However, the retrieval results can

be negatively affected (because LSH is an approximate search approach), and the LSH settings must be adjusted carefully to avoid a strong decrease in the retrieval quality. Examples of partition-based approaches include K-d trees [18, 61], which also have been used in MIR [118, 157, 216]. For example, McFee and Lanckriet compared several K-d tree variants in the context of music similarity search [118]. They found that the combination of a tree variant for approximate search, known as spill trees [109], with PCA [21] gives a favorable trade-off between accuracy and complexity. As an alternative, in our contribution, we explore a modern graph-based index structure called *hierarchical navigable small world* (HNSW) graph [114], which provides an approximate search solution, and already has been successfully used for, e.g., image retrieval [3].

In this chapter, we use the same cross-version music retrieval task as in Chapter 3 as an example application to explore the HNSW graph in practice. We conduct systematic experiments using databases of different sizes to analyze the impact of the graph-based index on the quality and speed of our retrieval system. As our main contribution, we show that we can increase the efficiency of our music retrieval application by several orders of magnitude through the usage of an HNSW graph as an index structure. Although the graph-based search belongs to the category of approximate nearest neighbor search approaches, our experiments show that, by and large, the retrieval quality is not negatively affected by using the graph in our scenario. We also analyze several further aspects in our retrieval system, such as the feature computation and the construction, saving, and loading of the index structure. We want to emphasize that our results demonstrate huge advantages of the graph-based indexing approach compared to previously used index structures. We expect similar benefits for other MIR problems involving nearest neighbor search. Beyond cross-version music retrieval, such search problems occur in diverse MIR tasks, such as cover song retrieval [49, 211], music similarity estimation [170], query-by-humming [164], or symbolic music genre classification [96].

To make our results reproducible [120], we use open-source implementations of the discussed index structures and provide code that shows how we use them, along with pre-computed features for an example dataset.[14] In this way, we enable a straightforward usage of the index structures in future MIR applications.

The remainder of this chapter is organized as follows. In Section 4.2, we outline our music retrieval application. Next, in Section 4.3, we describe graph-based search procedures in general and the HNSW graph in particular. Then, in Section 4.4, we present our systematic experiments where we apply the HNSW graph for our music retrieval task. Finally, we summarize our main findings in the concluding Section 4.5.

[14] `https://www.audiolabs-erlangen.de/resources/MIR/2020_signals-indexing`

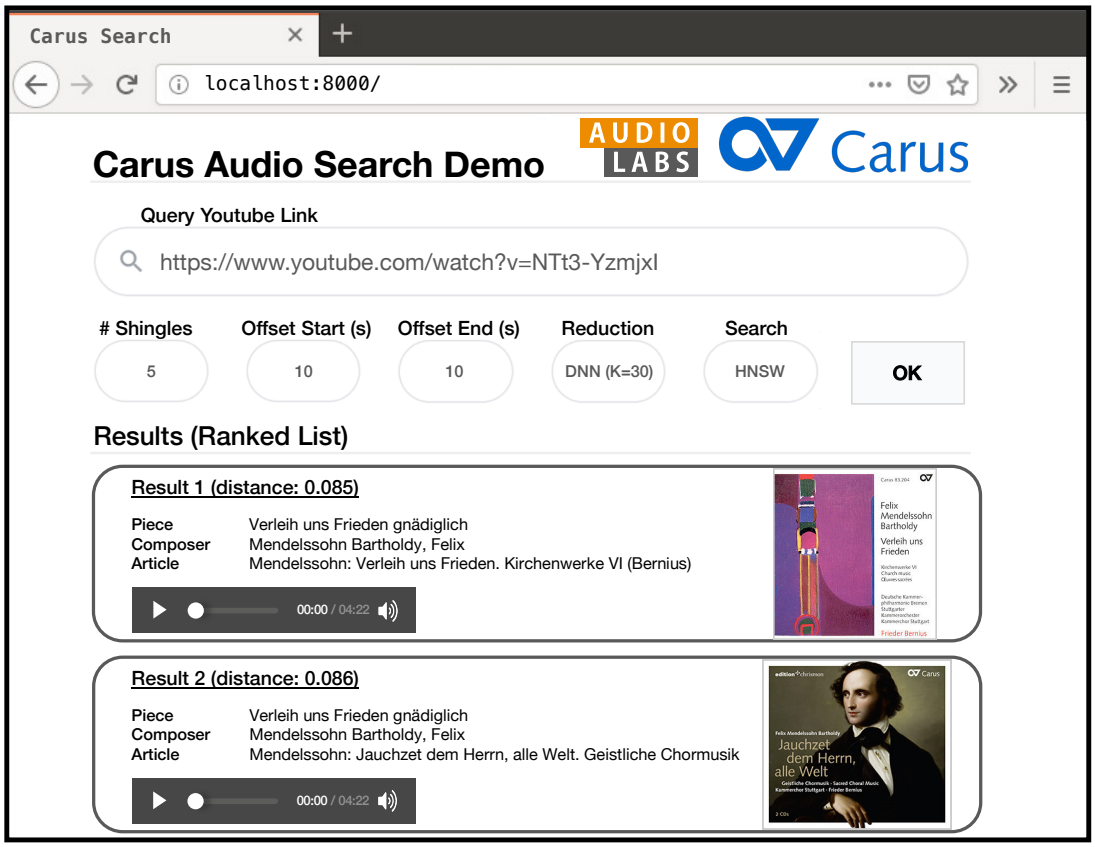


Figure 4.1: Illustrated screenshot of an internal version-retrieval web-interface for the Carus catalog.

4.2 Music Retrieval Application

In this section, we describe our motivating retrieval scenario, our cross-version retrieval approach, and our datasets, which are used later in our experiments. Readers with a primary interest in indexing and our experimental findings may skip this section at first reading.

4.2.1 Motivating Retrieval Scenario

In our study, we deal with a query-by-example retrieval scenario using a real-world music collection from a music publisher. This collection consists of the complete audio catalog of the Carus publishing house, a leading sheet music publisher of sacred and secular choral music. Beyond sheet music, Carus also produces and publishes audio recordings, mainly for choral pieces of Western classical music. Carus' complete audio catalog is a medium-sized music collection of nearly 400 hours (more details in Section 4.2.3). Internally, we implemented a web-based interface for browsing this dataset, as illustrated by the screenshot shown in Figure 4.1. Here, the user can specify a query in the form of a YouTube link of a music recording, e.g., an interpretation of Mendelssohn's *Verleih uns Frieden* (*Grant us peace*) by an amateur choir. Such

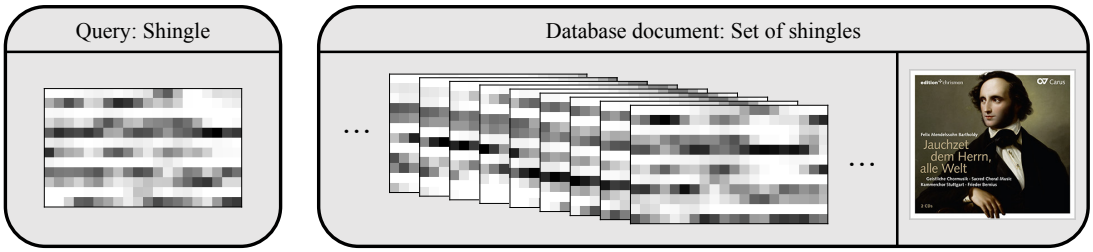


Figure 4.2: Illustration of shingle-based query and database document.

YouTube videos are often poorly annotated. Therefore, in our scenario, we use a content-based retrieval approach, where we take a 20-second audio snippet from the YouTube recording as a query. Then, the system retrieves recordings from the Carus collection that are based on the same musical piece as the query. Following [72, 216], we denote different recordings of the same piece of music as "versions." In the case of the Mendelssohn piece, the retrieval system returns two CD articles from the Carus catalog, which both include a version of that piece by professional musicians (the Kammerchor Stuttgart under the direction of Frieder Bernius). The user may listen to the retrieved versions or click on the cover images to access more information on the linked webpage of the publisher. Rather than describing this web-based tool in further detail, it serves as a motivating scenario for our retrieval experiments while indicating our study's practical relevance.

4.2.2 Cross-Version Retrieval System

In our study, we use the same retrieval approach as in the previous chapter (described in Section 3.3). For convenience, we shortly summarize the retrieval procedure. Given a database of music recordings and a short query audio fragment, the aim is to identify all recordings (versions) in the dataset based on the same musical piece as the query. To this end, we compare the database and query recordings using chroma-based audio features, which measure local energy distributions in the 12 chromatic pitch class bands [68, 128]. Our chroma features are computed by suitably pooling the frequency bins of a time–frequency representation with a logarithmic frequency axis, where a frequency bin corresponds to a semitone (originally introduced in [127], see Section 2.4 for more details). Following [72, 216], we use a chroma variant called CENS (chroma energy distribution normalized statistics) [132], which is adapted for the retrieval task using a post-processing strategy involving logarithmic quantization, temporal smoothing, and frame-wise normalization.

All database recordings are transformed into chroma sequences. We use a shingling approach [39, 40, 72, 216], where the database's chroma sequences are subdivided into subsequences (also referred to as "shingles") of $L = 20$ chroma vectors, using a hop size of $H = 1$ frame. The length of a shingle corresponds to 20 seconds of audio (using a feature rate of 1 Hz). As for the retrieval, the query (in the form of a single shingle) is compared with all database shingles. Figure 4.2 illustrates such a query (left) for our Mendelssohn example and the set of overlapping shingles (right) for a database document (a track of a

CD from the Carus collection). As the first option to compare two shingles, we reshape each shingle of dimension 12×20 to a vector of dimensionality $K = 240$ and apply a distance function

$$d\colon \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}_{\geq 0}\,. \tag{4.1}$$

Throughout this chapter, we use the squared Euclidean distance

$$d(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \sum_{k=1}^{K} (x_k^{(1)} - x_k^{(2)})^2 \tag{4.2}$$

between two vectors $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \in \mathbb{R}^K$ as the distance function.

Alternatively, we reduce the dimensionality ($K < 240$) of the shingles before computing the distance. In particular, as in Chapter 3, we use classical PCA [21], or DNNs trained with the triplet loss function [173] for dimensionality reduction (see Section 3.4 for more details). Our experiments of the previous chapter showed that the shingle dimension can be reduced from 240 to 15 without substantial loss in retrieval quality for the given application (while PCA- and DNN-based embeddings yield similar results). The DNN-based approach is beneficial for even smaller dimensionalities (below $K = 12$), where we only have a moderate loss in retrieval quality.

The retrieval task is then solved by finding the database's shingles with the smallest distance to the query shingle, which is a nearest neighbor search problem. In the previous chapter, we compared the runtimes for this retrieval task using an exhaustive search and a search approach using K-d trees [18, 61]. Using a small dataset of 16 hours of music, we found that K-d trees are only beneficial for smaller dimensionalities (below $K = 15$). This finding agrees with the fact that K-d trees are inappropriate for high-dimensional data [116, 157]. In this chapter, building upon these findings, we want to explore the potential of a graph-based index structure for this music retrieval problem.

4.2.3 Datasets

We use two databases of different sizes in our experiments (see Table 4.1). Our first set $\mathbb{D}_{\text{sm}}$, which was already used as an evaluation set in the previous chapter,[15] consists of 330 audio files and comprises about 16 hours of music (corresponding to more than 50 000 shingles). These recordings contain interpretations of orchestral and piano pieces by Beethoven, Chopin, and Vivaldi. In our cross-version retrieval evaluation, we consider a recording relevant for a query if it represents a version of the piece underlying the query (e.g., Mendelssohn's *Verleih und Frieden* or the first movement of Beethoven's Third Symphony). Since the dataset $\mathbb{D}_{\text{sm}}$ contains twelve "cliques" (works or movements), having a different number of versions each (4 to 67), a query may correspond to 4 to 67 relevant documents in our cross-version retrieval scenario.

[15] $\mathbb{D}_{\text{sm}}$ was denoted by $\mathbb{D}_2$ in the previous chapter. See Table 3.2 on p. 35 for more details. We rename the dataset because, in the context of this chapter, the dataset size (small or large) is our primary concern.

	# Audio Files	Σ Duration	⌀ Duration	Annotations	# Shingles
$\mathbb{D}_{\text{sm}}$	330	16:13:37	0:02:57	✓	52 332
$\mathbb{D}_{\text{lg}}$	7115	389:58:03	0:03:17	✗	1 272 386

Table 4.1: Statistics for the used datasets. Duration format hh:mm:ss. Annotations refer to the availability of complete annotations for the musical pieces underlying the recordings (required to evaluate the retrieval quality). Σ refers to the total, and ⌀ refers to the average.

This dataset is well annotated and corresponds to a controlled retrieval scenario. We make the annotations and shingles of the dataset available on our accompanying website for reproducibility.[14]

As a second dataset $\mathbb{D}_{\text{lg}}$, we use the entire audio catalog of recordings offered by the Carus label. The dataset $\mathbb{D}_{\text{lg}}$ contains 7115 audio files and about 390 hours of professionally produced music (corresponding to more than 1.25 million shingles), mainly of vocal Western classical music. In contrast to $\mathbb{D}_{\text{sm}}$, this dataset is less well annotated and corresponds to a rather uncontrolled scenario "in the wild," having real practical relevance.

4.3 Graph-Based Nearest Neighbor Search

We now outline the nearest neighbor search problem underlying our retrieval task and a search procedure using graph-based data structures [147]. Then, we describe the HNSW graph, which we use later as an index in our experiments.

4.3.1 Graph-Based Data Structures

In our search problem, we have a database $\mathbb{D}$ of items $\boldsymbol{x} \in \mathbb{D}$. The items are K-dimensional vectors, thus $\mathbb{D} \subset \mathbb{R}^K$. Figure 4.3 (initial situation) illustrates a dataset, where the items are gray points. Given a query $\boldsymbol{x}^{\text{q}} \in \mathbb{R}^K$ (colored in orange in Figure 4.3) and some distance measure d (see Equation 4.1), the aim is to find the $\nu \in \mathbb{N}$ database items that are nearest to this query. As an example, let us now assume $\nu = 1$. Then the aim is to find the closest database item

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathbb{D}}{\arg\min}\, d(\boldsymbol{x}, \boldsymbol{x}^{\text{q}}). \tag{4.3}$$

A naive solution to this problem is to compute the distances between all database items and the query for selecting the item with the smallest distance (exhaustive search). With graph-based nearest neighbor search, we aim to find $\boldsymbol{x}^*$ without evaluating all these distances, but only a subset of them. In general, we have no guarantee of finding $\boldsymbol{x}^*$ with a graph-based search. Because of this reason, this strategy belongs to the category of approximate nearest neighbor search methods.

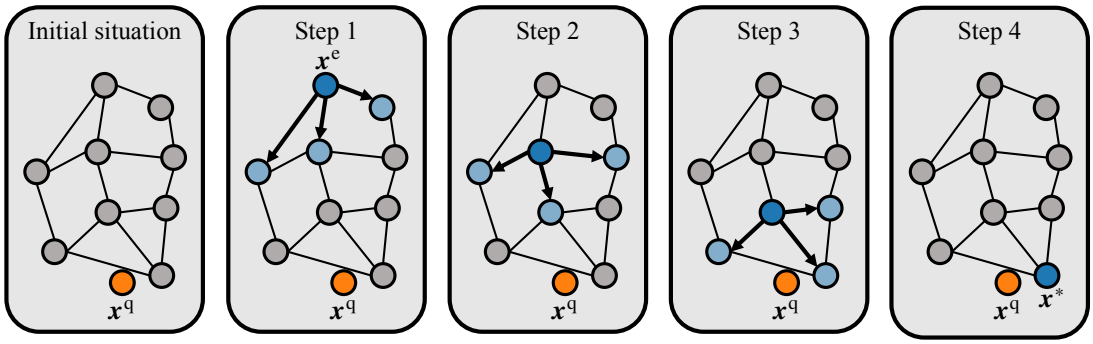


Figure 4.3: Illustration of graph-based approximate nearest neighbor search.

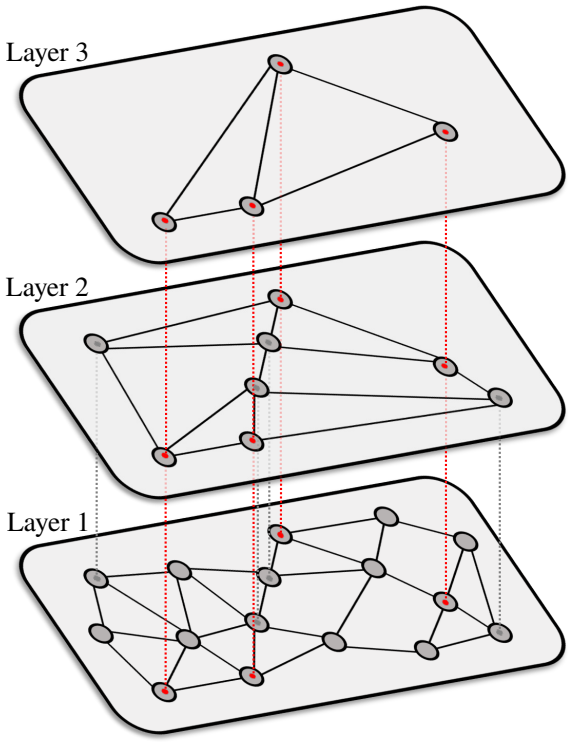
In our graph-based approach, the database is organized as an undirected graph structure, where the database items are nodes. Nearby nodes are connected by edges, which are represented by connecting lines in Figure 4.3. In essence, the search in the database consists in traversing along the edges of the graph. In the first step of the search procedure, we select an entry point $\boldsymbol{x}^{\mathrm{e}} \in \mathbb{D}$ of the database (colored in dark blue in Figure 4.3, first step), e.g., by random choice. This entry point is our first active search node in the procedure. We then compute the distance of the query to this node. Next, we compute the distance between the query and all items that are connected to the active search node (colored in light blue in Figure 4.3). If any of these distances are smaller than the distance between the query and the active search node, we continue with the next step, where the node with the shortest distance is the next active search node in the procedure. Otherwise, we terminate, and our active search node is the final candidate for the nearest neighbor to the query. In Figure 4.3, we perform four steps until the algorithm terminates. In our example, the final candidate corresponds to the exact solution $\boldsymbol{x}^*$.

4.3.2 HNSW Graphs

So far, we have described a search procedure using a data structure with a single graph. Building upon such a structure, Malkov and Yashunin [114] introduced an improved data structure with multiple levels, called HNSW graph, for approximate nearest neighbor search. Compared to search procedures for single-layer structures (as described in the previous section), the multi-layer search procedure has various benefits, such as improved search quality, higher efficiency (with a runtime that increases logarithmically with the dataset size), and greater stability with respect to the dimensionality K.

Figure 4.4 illustrates an HNSW graph with three layers. The bottom level (first layer) is a graph containing the full database, similar to the graph structure described in the previous section. In the figure, this layer contains 16 database items. The middle level (second layer) contains a subset of these items (eight items). The top level (third layer) contains a subset of the second layer's items (four items). For illustration purposes, the dashed red lines in the figure indicate the items that are available at all layers. Items that are available only at the first two layers are indicated with dashed gray lines.

Figure 4.4: Illustration of HNSW graph.



We now want to outline the search procedure for HNSW graphs. Given a query point $\boldsymbol{x}^{\mathrm{q}}$, searching in an HNSW graph starts at the top layer with a suitably selected entry point $\boldsymbol{x}^{\mathrm{e}}$ (e.g., random selection, see [114] for details). A search procedure is then applied, similar to the approach described in the previous Section 4.3.1, to find a candidate for the nearest neighbor to $\boldsymbol{x}^{\mathrm{q}}$ in the top layer. Then, the search continues at the next layer, where the entry point is the item corresponding to the nearest neighbor candidate from the upper layer. This way, the search continues until we arrive at the bottom layer, where all database items are available. The aim is to find the ν nearest neighbors in this layer as the search procedure's result. To stabilize the approximate search results, we may first search for more than ν approximate nearest neighbors (using a graph-based search procedure as before). We denote the number of "intermediate" candidates by $\nu' \in \mathbb{N}$, where $\nu \leq \nu' \leq |\mathbb{D}|$. Among the candidates, we then select the ν nearest neighbors (by exhaustive search) as the final result. The number ν' is a free parameter that can be increased to improve the search results (at the cost of an increased runtime).

Malkov and Yashunin [114] also proposed an algorithm for constructing HNSW graphs, which we summarize briefly. During the construction process, the database items are consecutively inserted into the graph. For each new item $\boldsymbol{x} \in \mathbb{D}$, we randomly decide on its upper-most layer $\ell(\boldsymbol{x}) \in \mathbb{N}$ according to an exponentially decaying probability distribution. If the random process selects a higher layer $\ell(\boldsymbol{x})$ than the highest existing layer in the graph, the number of layers in the graph increases dynamically. The item $\boldsymbol{x}$ is then inserted in all layers $[1 : \ell(\boldsymbol{x})] := \{1, \ldots, \ell(\boldsymbol{x})\}$. Next, we want to connect the new node $\boldsymbol{x}$ to M database items in each layer $[1 : \ell(\boldsymbol{x})]$ by edges. The parameter $M \in \mathbb{N}$ controls the minimum number of

edges for each node of the data structure. We apply a top-down search procedure, similar to the approach described above, to search for suitable items in each layer $[1 : \ell(\boldsymbol{x})]$. To stabilize the search results, we may first search for more than M candidates for inserting edges (using a graph-based search procedure as before). Similar to the number ν' of intermediate neighbor candidates in the search procedure, we denote the number of intermediate candidates for inserting edges by $M' \in \mathbb{N}$. We select M database items among the M' candidates for inserting edges. In their paper [114], the authors propose two options for this selection. As a first option, they select the nearest neighbors to $\boldsymbol{x}$ by exhaustive search. As a second option, the authors propose a heuristic to create connections to $\boldsymbol{x}$ from diverse directions, which is beneficial for highly clustered data (see the original paper [114] for more details). In our experiments, we use the second option. If any of the nodes turn out to have more edges than a predefined maximum (set to $2M$ for the bottom layer and M for all other layers), the nodes' surplus edges with the highest distance are removed.

In summary, we have various important parameters for the construction and search procedure of an HNSW graph. Beyond the distance function d, these parameters are ν (number of neighbors to search for), ν' (number of neighbor candidates during search), M (minimum number of edges for each node), and M' (number of edge candidates during construction). The authors recommend a range of $M \in [5 : 48]$, where higher M values imply better search results and higher memory consumption. Furthermore, they recommend increasing M for higher dimensionalities K. In our experiments described in the next section, we fix the squared Euclidean distance (see Equation 4.2) as distance function d as well as the default parameter settings of $\nu' = M' = 100$ and $M = 5$. As we will see later, a fine-tuning of these parameters is not necessary for obtaining good results in our application.

4.4 Experiments

We now use the HNSW graph as an index structure in our music retrieval application. Here, a node of the graph corresponds to a database shingle with or without dimensionality reduction.

4.4.1 Experimental Setup

In the following, we analyze the possible decrease in retrieval quality and the improvements of the retrieval runtime introduced by the HNSW graph. To this end, we consider quantitative performance measures, which we list in Table 4.2. To evaluate the retrieval quality, we use standard precision-based measures (more details in Section 4.4.2). To analyze the impact of the index structures on the retrieval speed, we consider several steps that are involved in our retrieval scenario. Some steps need to be computed offline when processing the database documents, and other steps need to be computed online when processing a query. In the offline phase, we need to compute features for all audio files of the database, construct an index and save the index file to disk. These steps can be carried out at any time and on any system (offline). When applying our index, we first need to load the index into the computer's main memory (RAM). This

Table 4.2: Considered performance measures.

Step	Performance Measure	Stage	Section
Overall pipeline	Quality (P@1, P@3, P_R)	Offline & online	4.4.2
Feature computation	Time (ms)	Offline & online	4.4.3
Constructing the index	Time (s)	Offline	4.4.4
Saving the index	Disk space (MB)	Offline	4.4.4
Loading the index	Time (ms)	Offline	4.4.4
Retrieval	Time (ms)	Online	4.4.5

loading step can be considered as being in-between the online and offline stages. The index loading needs to be performed on the actual system where the retrieval service is provided. When the index structure can be kept in the main memory, it does not have to be reloaded for each query. Therefore, we still consider it as part of the offline stage. In the actual online phase, we need to compute the query features and perform the nearest neighbor search procedure using our index structure. In the following sections, we analyze these steps in the order given in Table 4.2.

If not mentioned otherwise, we always report on average time measurements ($\mu \pm \sigma$) for 100 iterations of the experiment. Note that runtime evaluation is a delicate topic on its own [98]. For example, one may argue that it is more meaningful to report minimum instead of average runtime measurements because other processes running in parallel affect the mean more than the minimum. In our case, this is not a major issue because the standard deviation σ is always relatively low. We want to highlight that we take a practical perspective by measuring runtimes using distinct implementations of the respective index structures, implemented in different programming languages. The absolute runtimes obtained may vary when using different implementations or hardware systems. Our study gives practical insights into the runtimes obtained by specific implementations on specific platforms for our specific application. In general, we are interested in the orders of magnitude, the relative differences between the time measurements, and the relationships between index size and runtime.

We compare three different search approaches: an exhaustive search approach (full search), an indexing strategy using K-d trees (KD), and the graph-based index structure (HNSW). We perform our experiments using Python 3.6.5 on a computer with an Intel Xeon CPU E5-2620 v4 (2.10 GHz) and 31 GiB RAM. We use the efficient pairwise-distance calculation of scipy 1.0.1 [199] for the full search, which calls a highly optimized implementation in C. For the K-d trees (using a default leaf size of 30), we use the implementation of scikit-learn 0.20.1 [142], which is written in Cython. For the HNSW graph, we use the efficient hnswlib implementation in C++ by the authors of the original paper [114][16] (using the Python wrapper version 0.4.0). We use librosa 0.7.1 [119] for the audio processing pipelines and TensorFlow 1.7.0 [2] for the DNN implementation.

[16] `https://github.com/nmslib/hnswlib`

Reduction	K	Search	P@1	P@3	P_R
—	240	Full Search / KD	1.0000	0.9965	0.9434
		HNSW	1.0000	0.9965	0.9434
PCA	30	Full Search / KD	1.0000	0.9910	0.9130
		HNSW	1.0000	0.9910	0.9130
	12	Full Search / KD	1.0000	0.9679	0.8294
		HNSW	1.0000	0.9678	0.8294
	6	Full Search / KD	1.0000	0.8937	0.7350
		HNSW	1.0000	0.8937	0.7350
DNN	30	Full Search / KD	1.0000	0.9868	0.9344
		HNSW	1.0000	0.9869	0.9345
	12	Full Search / KD	1.0000	0.9757	0.8989
		HNSW	1.0000	0.9756	0.8989
	6	Full Search / KD	1.0000	0.9236	0.8333
		HNSW	1.0000	0.9237	0.8333

Table 4.3: Retrieval quality for various reduction approaches and search strategies using the dataset $\mathbb{D}_{\text{sm}}$.

4.4.2 Retrieval Quality

In contrast to the K-d tree approach, the HNSW graph only provides an approximate search solution. To understand the impact of this approximation within our retrieval scenario, we measure our retrieval system's quality using the dataset $\mathbb{D}_{\text{sm}}$, as done in Chapter 3. For convenience, we now summarize our evaluation approach.

We consider a document-level rather than a shingle-level retrieval. Here, the distance between a query shingle and a document is given by the minimizing distance between the query and all document shingles. We construct a single index structure for the entire dataset $\mathbb{D}_{\text{sm}}$ (using either a K-d tree or an HNSW graph) and search for the 10 000 nearest items in the database to a given query. Using the distances of the returned items, we create a ranked list of documents, ordered by ascending distances. Note that we were not able to rank all database documents as some documents may not have a corresponding item among the items returned (this did not affect the evaluation measures in our experiments). For evaluating the ranked list, we consider three standard retrieval evaluation measures [115]. First, we use precision at one (P@1), which is 1 if the top-ranked document is relevant, and 0 otherwise. Note that, for exact nearest neighbor searches, the top match is always identical to the query because, in our experiments, the query is part of the database (unlike in the experiments of the previous chapter). We still use this measure to check whether the approximate search approach is able to find the "trivial" match. Second, we use precision at three (P@3), which is the proportion of relevant documents among the top 3 documents of the ranked list. Third, we use R-precision (P_R), which is the proportion of relevant documents among the first R ranks, where $R \in \mathbb{N}$ denotes the number of relevant documents for the given query (which may differ for each query, between 4 and 67).

We generate 3300 query shingles from $\mathbb{D}_{\text{sm}}$ by an equidistant sampling of ten queries from each recording of $\mathbb{D}_{\text{sm}}$, resulting in 3300 queries. Each evaluation measure is finally averaged over the 3300 query shingles used. Table 4.3 shows the resulting evaluation measures. A row in this table specifies the dimensionality reduction approach (no reduction, PCA-, or DNN-based embedding), the dimensionality K, and the search strategy (Full Search, KD, or HNSW). The retrieval results for the exhaustive search (Full Search) and the K-d tree strategy (KD) are identical because both approaches are exact nearest neighbor searches. For example, without dimensionality reduction, we obtain a P@1 value of 1.0, a P@3 value of 0.9965, and a P_R value of 0.9434. This result shows that the shingle-based retrieval approach is able to identify most of the versions correctly, but there are a few false positives. Reducing the shingle dimensions (which is important for some indexing approaches, such as K-d trees) leads to further degradations of the retrieval quality, as already shown in the previous chapter. For example, reducing the dimensionality from 240 to 30 with the PCA-based embedding, we obtain a P@3 value of 0.9910. For smaller dimensionalities, the DNN is beneficial over PCA for embedding the shingles, e.g. resulting in P_R values of 0.7350 (PCA) and 0.8333 (DNN) for $K = 6$. Using the HNSW graph as an index structure, we obtain more or less the same evaluation metrics for all settings. This finding demonstrates that the approximate search approach of the HNSW graph has almost no negative impact on the retrieval results within our application scenario. When we analyze the runtime improvements in the following sections, we can bear in mind that they come without substantial loss in retrieval quality.

4.4.3 Feature Computation

In this section, we report on the runtimes for the various steps involved in the feature computation. This computation procedure has to be performed in the offline stage (for the whole database) and online stage (for the query). In contrast to the document-based analysis of the retrieval quality (evaluating a ranked list of documents), we now use an item-based evaluation (runtime to process a database item, i.e., a shingle or a shingle embedding). To compute our measurements, we first load 20 seconds of an audio file (using `librosa.load`). In general, our audio files are longer, but for our runtime experiments, we only use a 20-second segment, corresponding to the length of a shingle. Then, we compute the spectral features [127] (with `librosa.iirt`). Next, we compute the CENS features [132] (using `librosa.feature.chroma_cens`). This step concludes the feature computation if no dimensionality reduction is applied. An additional step is performed in the embedding-based retrieval approaches (PCA-based embedding using `sklearn.decomposition.PCA` or DNN-based embedding as described in the previous chapter).

Table 4.4 shows the time measurements. The loading of the audio segment only takes 44.8 ms. The next step is the spectral feature computation, which needs more than a second (1171.5 ms). The time required for the CENS computation is not significant (0.9 ms). In the table, we list the times to embed the shingle for some selected dimensionalities. In general, the PCA-based embedding is faster than 0.1 ms, and the DNN-based embedding does not take more than 2 ms.

Step	Time (ms)
Audio Loading	44.8 ± 1.4
Spectral Feature Computation	1171.5 ± 34.3
CENS Feature Computation	0.9 ± 0.5
Embedding PCA ($K = 30$)	0.05 ± 0.0
Embedding PCA ($K = 12$)	0.05 ± 0.0
Embedding PCA ($K = 6$)	0.05 ± 0.0
Embedding DNN ($K = 30$)	1.6 ± 0.1
Embedding DNN ($K = 12$)	1.5 ± 0.1
Embedding DNN ($K = 6$)	1.4 ± 0.1

Table 4.4: Time measurements (in ms) for various steps involved in the feature computation of 20 seconds of audio.

The numbers of the table show that the major bottleneck of the feature computation is the spectral transform. Compared to this, the times of the other steps are not significant. The runtime for the query feature transform is not our focus in this study. Obviously, it only scales linearly with the query length, which is usually short (i.e., no scalability issue). The runtime for computing the database documents' features is also not critical because it can be computed offline. A possible future research direction could be to compute embeddings from spectral representations that are less expensive to compute, e.g., using the STFT with the FFT, opposed to the IIR-based log-frequency spectrogram used here (see Section 2.4). Having the same window and hop length settings as the spectral transform used, computing the magnitude STFT for the 20-seconds audio snippet only takes 19.2 ms on average. However, using the STFT-based features may go along with a decrease of feature quality, which may affect the retrieval results.

4.4.4 Constructing, Saving, and Loading the Index

We now address various performance measures for the offline stage, i.e., for constructing, saving, and loading the index structures. For these steps, we restrict our analysis to one embedding technique (PCA) because the specific embedding strategy used has only a minor influence on these measures. The first step is to construct the index structure (either a K-d tree or an HNSW graph). We report on the times required to construct the index, given that the data to be indexed is already in the computer's main memory (i.e., pre-computed shingle embeddings, without having any distances pre-computed). When this data needs to be read from disk, it will cause some additional overhead. For example, loading all pre-computed shingle embeddings ($K = 12$) of $\mathbb{D}_{\text{sm}}$ takes 0.9 ms on average. Loading the full shingles ($K = 240$) of $\mathbb{D}_{\text{lg}}$ requires one second on average.

Columns 4 and 5 of Table 4.5 show the time needed to construct the index structures for various dimensionalities. We include time measurements for the smaller dataset $\mathbb{D}_{\text{sm}}$ and the larger dataset $\mathbb{D}_{\text{lg}}$. The first row in the table refers to the K-d tree index for shingles without dimensionality reduction ($K = 240$). This setting leads to construction times of 0.54 s for $\mathbb{D}_{\text{sm}}$ and 43.03 s for $\mathbb{D}_{\text{lg}}$. For lower dimensions, this time decreases. For example, constructing the K-d tree index for $\mathbb{D}_{\text{lg}}$ takes 3.89 s, 1.66 s,

Search	Reduction	K	Construction Time (s)		Save Size (MB)		Load Time (ms)	
			$\mathbb{D}_{\text{sm}}$	$\mathbb{D}_{\text{lg}}$	$\mathbb{D}_{\text{sm}}$	$\mathbb{D}_{\text{lg}}$	$\mathbb{D}_{\text{sm}}$	$\mathbb{D}_{\text{lg}}$
KD	—	240	0.54 ± 0.0	43.03 ± 0.1	209.7	5161.2	131.5 ± 0.4	3209.1 ± 25.4
KD	PCA	30	0.06 ± 0.0	3.89 ± 0.2	27.0	664.8	17.5 ± 0.3	405.1 ± 8.8
KD	PCA	12	0.03 ± 0.0	1.66 ± 0.1	11.3	279.4	3.5 ± 0.1	167.0 ± 6.9
KD	PCA	6	0.02 ± 0.0	0.99 ± 0.1	6.1	150.9	2.1 ± 0.3	86.3 ± 2.2
HNSW	—	240	0.65 ± 0.0	26.66 ± 0.0	53.9	1310.9	108.1 ± 0.4	2680.0 ± 42.8
HNSW	PCA	30	0.51 ± 0.0	16.38 ± 0.0	9.9	241.8	85.1 ± 2.1	2095.3 ± 6.6
HNSW	PCA	12	0.43 ± 0.0	11.58 ± 0.0	6.2	150.2	82.9 ± 0.4	2071.5 ± 43.8
HNSW	PCA	6	0.42 ± 0.0	10.20 ± 0.0	4.9	119.6	82.6 ± 0.6	2049.1 ± 22.8

Table 4.5: Performance measures for constructing, saving, and loading the index structure.

and 0.99 s for the dimensionalities of 30, 12, and 6, respectively. Constructing an HNSW graph for $K = 240$ requires 0.65 s for the smaller dataset $\mathbb{D}_{\text{sm}}$ and 26.66 s for the larger dataset $\mathbb{D}_{\text{lg}}$. For this large dataset and a high dimensionality of $K = 240$, constructing an HNSW graph is faster (26.66 s) than constructing a K-d tree (43.03 s) in the implementations used. However, this is not the case for lower dimensions. For example, constructing the index structures for the larger dataset $\mathbb{D}_{\text{lg}}$ using $K = 30$ requires 3.89 s for the KD and 16.38 s for the HNSW approach. In general, the construction time grows approximately in a linear fashion with the dimensionality K for lower dimensions. Only for large dimensionalities, the time for constructing a K-d tree explodes, which agrees with the fact that K-d trees are not suited for high-dimensional data [116, 157]. In contrast, the HNSW approach behaves stable for all considered dimensionalities. In all our settings, constructing an index takes less than a minute. We do not consider this duration critical in our application because the step is performed offline.

The next step is to save the index structure to the hard disk, where it requires disk space. In the case of the K-d tree, we use scikit-learn's [142] recommended default persistence format based on the Python package joblib without compression. In the case of the HNSW graph, we use the default storage format of hnswlib, which is a custom binary format (also without compression). Columns 6 and 7 of Table 4.5 show the required disk space used for storing the index structures. Without dimensionality reduction ($K = 240$), storing the K-d tree requires 209.7 MB and 5161.2 MB of disk space for $\mathbb{D}_{\text{sm}}$ and $\mathbb{D}_{\text{lg}}$, respectively. The HNSW graph takes 53.9 MB and 1310.9 MB for the same data. In general, the required disk space scales roughly linearly with the dataset size as well as with the dimensionality in both indexing approaches. Furthermore, the graph-based index is generally more space-efficient than the tree-based structure in the given formats.

To apply a pre-computed index for retrieval, we need to load it into the computer's main memory. We perform this step of loading the index with the functions required for the respective file formats used in the previous step (using functions from joblib and hnswlib, respectively). Columns 8 and 9 of Table 4.5 show the required time to load the index files. Loading a K-d tree without dimensionality reduction requires 131.5 ms for $\mathbb{D}_{\text{sm}}$ and 3209.1 ms for $\mathbb{D}_{\text{lg}}$. Using the same data, loading an HNSW graph takes 108.1 ms and 2680.0 ms, respectively. For smaller dimensions, loading a K-d tree is faster than loading an HNSW

graph (e.g., $K = 12$ and $\mathbb{D}_{\text{lg}}$: 167.0 ms for KD and 2071.5 ms for HNSW). The time to load an index scales linearly with the dimensionality in both KD and HNSW approaches (with a much flatter slope for HNSW).

With our system (see Section 4.4.1 for specifications), we did not have any issues with loading the index structures into the main memory (31 GiB RAM) in all settings. On other systems with less memory (8 GiB RAM), we could not fully load the index structures into the main memory for $K = 240$. In this case, dimensionality reduction becomes crucial for practical reasons.

4.4.5 Retrieval Time

In this section, we report on our runtime experiments for searching the nearest neighbors in our datasets. Regarding efficiency, these experiments refer to the most critical part of the retrieval pipeline because this step is part of the online stage (where the runtime affects the user experience). An efficient search is of major importance for scalability because, in general, the search runtime scales with the dataset size. The aim of our index structures is to improve the efficiency of this step.

We can also consider the complexity of the search approaches from a theoretical perspective (which is not the focus of this chapter, being a practice report). The runtime of the full search increases linearly with the dataset size. In other words, its search complexity is in the order of $O(|\mathbb{D}|)$. The expected search complexity for K-d trees is in the order of $O(\log |\mathbb{D}|)$ [61]. However, it is well known [155] that the actual performance may be equivalent or worse than an exhaustive search, depending on the data distribution (which influences the tree structure). Especially for high-dimensional data, the search performance degenerates. According to [114], the overall complexity scaling of the search for the HNSW graph is $O(\log |\mathbb{D}|)$, which does not degenerate for high-dimensional data.

For the experiments of this section, we search for the ν nearest items in our dataset (either $\mathbb{D}_{\text{sm}}$ or $\mathbb{D}_{\text{lg}}$) to a given query, where we consider $\nu \in \{1, 10, 100, 1000\}$. We again use 3300 queries (as in Section 4.4.2) and perform several repetitions of this experiment. Then, we normalize the measured runtimes with respect to the number of queries and repetitions, such that the reported measures refer to the time needed for a single query. Table 4.6 shows the results of our experiments. Let us first consider the PCA-based dimensionality reduction using $K = 30$ for the smaller dataset $\mathbb{D}_{\text{sm}}$. The exhaustive search requires 4.803 ms. The indexing approaches are much faster, taking 0.009 ms using a K-d tree and 0.007 ms using an HNSW graph. In this setting, there is no large difference between the indexing approaches. When we search for more neighbors (increasing ν), the KD approach slows down substantially (0.621 ms, 1.001 ms, and 2.025 ms for ν values of 10, 100, and 1000, respectively). The runtime does not increase to the same extent for the HNSW approach (0.007 ms, 0.007 ms, and 0.056 ms). Searching in the larger dataset $\mathbb{D}_{\text{lg}}$ shows the potential of the HNSW index even more clearly. For example, using the PCA-based embedding ($K = 30$), for $\nu = 100$, the K-d tree requires 66.499 ms and the HNSW graph needs only 0.019 ms. While the increased dataset size only has a minor effect on the runtime for the HNSW approach, it dramatically increases the KD strategy's runtime. For lower dimensionalities, the runtime differences between the K-d

Reduction	K	Search	$\mathbb{D}_{\text{sm}}$ $\nu = 1$	$\nu = 10$	$\nu = 100$	$\nu = 1000$	$\mathbb{D}_{\text{lg}}$ $\nu = 1$	$\nu = 10$	$\nu = 100$	$\nu = 1000$
—	240	Full Search	13.912	13.912	13.824	13.875	—	—	—	—
		KD	0.072	23.136	24.463	25.269	771.623	775.116	770.461	772.946
		HNSW	0.008	0.008	0.009	0.072	0.020	0.020	0.021	0.205
PCA	30	Full Search	4.803	4.757	4.768	4.812	—	—	—	—
		KD	0.009	0.621	1.001	2.025	36.825	48.145	66.499	88.849
		HNSW	0.007	0.007	0.007	0.056	0.010	0.009	0.019	0.123
PCA	12	Full Search	4.354	4.418	4.322	4.360	—	—	—	—
		KD	0.004	0.087	0.233	0.713	0.907	1.698	4.414	10.280
		HNSW	0.007	0.007	0.007	0.056	0.009	0.009	0.009	0.067
PCA	6	Full Search	4.123	4.115	4.124	4.159	—	—	—	—
		KD	0.003	0.019	0.074	0.346	0.031	0.065	0.201	0.852
		HNSW	0.007	0.007	0.006	0.048	0.007	0.007	0.011	0.072
DNN	30	Full Search	4.899	4.897	4.883	4.930	—	—	—	—
		KD	0.009	0.459	1.019	2.498	30.021	46.611	75.748	117.543
		HNSW	0.007	0.007	0.007	0.057	0.011	0.011	0.011	0.077
DNN	12	Full Search	4.439	4.434	4.433	4.480	—	—	—	—
		KD	0.004	0.064	0.177	0.709	0.377	0.829	2.498	7.240
		HNSW	0.008	0.008	0.009	0.068	0.006	0.007	0.007	0.056
DNN	6	Full Search	4.165	4.162	4.171	4.220	—	—	—	—
		KD	0.002	0.014	0.062	0.310	0.019	0.039	0.141	0.663
		HNSW	0.006	0.007	0.007	0.054	0.007	0.007	0.007	0.057

Table 4.6: Search runtimes (in ms) using a single query for various dimensions K and search strategies.

tree and the HNSW graph are less extreme. For example, the KD approach requires 0.201 ms for $K = 6$ ($\mathbb{D}_{\text{lg}}$, $\nu = 100$). Still, the HNSW graph is much faster (0.011 ms). Without dimensionality reduction, the KD approach breaks down (more than 700 ms for $\mathbb{D}_{\text{lg}}$), which is a known fact [116, 157]. However, the HNSW graph still facilitates fast retrieval (e.g., 0.021 ms for $\nu = 100$). This substantial decrease in retrieval time shows the power of the graph-based search approach. In general, the tendencies discussed for the PCA reduction are similar when using the DNN-based embedding.

Note that the exhaustive search involves computing all pairwise distances between the query and the database items. As a consequence, the parameter ν does not influence the search time. Furthermore, we did not perform the full search for $\mathbb{D}_{\text{lg}}$ because of excessive memory requirements.

Figure 4.5 shows the search runtimes for various dimensionalities K, where the solid lines show μ and the light areas show $\pm\sigma$ around μ for the repetitions of our experiment. We observe that the runtime increases more than linearly with dimensionality K for the K-d tree. In contrast, the runtime for the HNSW graph increases only slightly with increasing dimensionality K. Note the different scales of the vertical axes, which again underline the substantial search time improvements caused by the HNSW graph. We see that the HNSW approach requires nearly the same time for searching 1, 10, or 100 database items (resulting in overlapping curves). The reason for this is the parameter setting $\nu' = 100$ (number of intermediate neighbor candidates, described in Section 4.3.2), which leads us to search internally for 100 neighbors anyway.

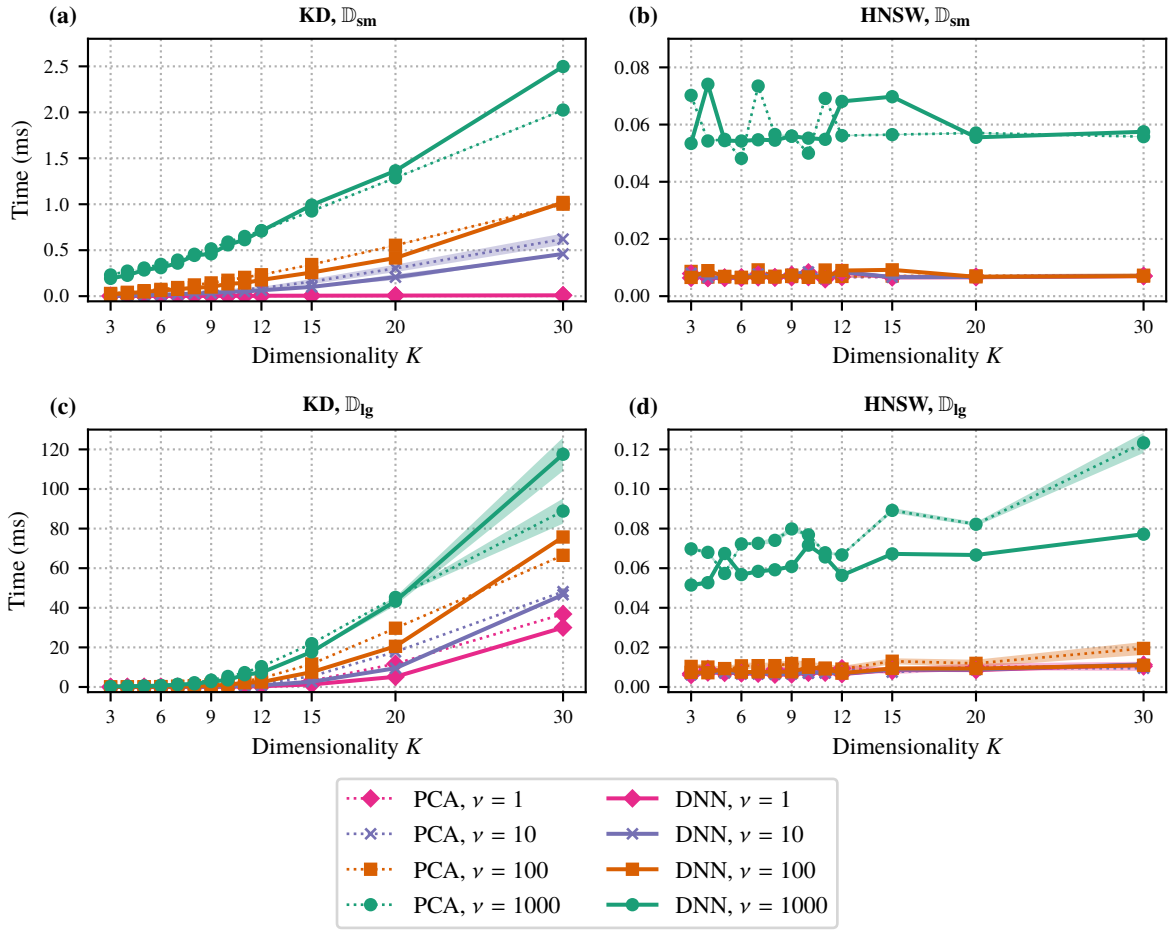


Figure 4.5: Search runtimes (in ms) for the DNN- and PCA-based embedding approaches. Note the different scale of the vertical axes. Search using **(a)** the KD strategy and $\mathbb{D}_{sm}$, **(b)** the HNSW strategy and $\mathbb{D}_{sm}$, **(c)** the KD strategy and $\mathbb{D}_{lg}$, and **(d)** the HNSW strategy and $\mathbb{D}_{lg}$.

To summarize, we can conclude that the dimensionality reduction approach (PCA or DNN) has only a minor influence on the runtime, which is expected. The dimensionality K of the index items has a substantial impact on the runtime. Still, the indexing approach (KD or HNSW) has the most important effect on the runtime. Our experiments show that, compared to K-d trees, the HNSW index is much faster and more stable concerning the dimensionality and the number of items to be indexed. This increase in retrieval efficiency comes without substantial loss in retrieval quality (as shown in Section 4.4.2), which makes the HNSW graph a powerful tool for music retrieval.

4.5 Conclusions

In our study, we compared various search techniques for a cross-version music retrieval task, where we aim to find the closest shingles in a database to a given query shingle. In particular, using datasets of different sizes, we applied indexing approaches with exact (exhaustive search, K-d tree) and approximate

(HNSW graph) solutions. Our results showed that the approximate solution of the graph-based index has almost no negative impact on the retrieval quality in our music scenario. As our main finding, we obtained dramatic speed-ups by several orders of magnitude for the search operations involved in our retrieval system. We verified that HNSW graphs are robust with respect to the dimensionality of the items to be indexed, unlike K-d trees. As another contribution, we explored the impact of the HNSW index on several further steps in our pipeline, such as constructing, saving, and loading the index structure.

This chapter was based on a previous study (Chapter 3) that used shingles with highly specialized features for a cross-version music retrieval task. We aimed to reduce the shingle dimensionality with different embedding strategies to make the retrieval application more efficient. As a main result, we found that it is possible to substantially reduce the shingle dimensionality with only a moderate loss in retrieval quality, where a DNN-based embedding is beneficial over a PCA-based reduction for small dimensionalities below $K = 12$. This reduction in dimensionality was essential for using K-d trees. Our experiments with HNSW graphs demonstrated that the shingle dimensionality is not as relevant to efficiency as good indexing approaches. Still, dimensionality reduction may be important, e.g., to reduce disk space and memory requirements. Given our results, the remaining bottleneck of the retrieval pipeline is the feature computation for the query. In future research, one may employ feature representations that are less expensive to compute, e.g., using the STFT. Here, the embedding techniques may be useful to adapt and further enhance these "raw" representations for the retrieval task.

As for future work, one may also explore alternative approaches to accelerate the nearest neighbor search in our cross-version retrieval scenario. One possibility is to apply an appropriate prototype selection method [63, 159], where the dataset is reduced by selecting representative prototypes. Another option is the use of pivot-based methods [31, 32], where pre-computed distances between the database items to some selected items (the pivots) are exploited for excluding some database items during the search.

As basis for further research in this direction, we make our work reproducible by using open source implementations in all steps and by providing example code that shows how to apply these implementations, along with feature representations for an example dataset.[14] Given our strong improvements in retrieval runtime without quality loss, we consider the HNSW graph a powerful tool that deserves more attention in the MIR community.

Part II

Learning Theme-Based Salience Representations

5 MTD: A Multimodal Dataset of Musical Themes for MIR Research

This chapter is based on [222]. As the first author and main contributor, Frank Zalkow developed the ideas and wrote the dataset article in collaboration with his supervisor Meinard Müller. Several people have contributed to the development of the dataset. Major contributions to data preparation and annotations stem from Frank Zalkow, Stefan Balke, Vlora Arifi-Müller, and Meinard Müller, who were assisted by several student associates, in particular Lena Krauß, Lukas Lamprecht, Anna-Luisa Römling, and Quirin Seilbeck.

Musical themes are essential elements in Western classical music. In this chapter, we present the musical theme dataset (MTD), a multimodal dataset inspired by "A Dictionary of Musical Themes" by Barlow and Morgenstern from 1948. For a subset of 2067 themes of the printed book, we created several digital representations of the musical themes. Beyond graphical sheet music, we provide symbolic music encodings, audio snippets of music recordings, alignments between the symbolic and audio representations, as well as detailed metadata on the composer, work, recording, and musical characteristics of the themes. In addition to the data, we also make several parsers and web-based interfaces available to access and explore the different modalities and their relations through visualizations and sonifications. These interfaces also include computational tools, bridging the gap between the original dictionary and MIR research. The dataset is of relevance for various subfields and tasks in MIR, such as cross-modal music retrieval, music alignment, optical music recognition (OMR), music transcription, and computational musicology.

5.1 Introduction

Western classical music is largely based on musical themes [54]. Such a theme is a musical idea used to build a composition (or a part of it). Often, this idea is a prominent melody that is announced in the first measures and is easily recognizable by the listener. Throughout the musical piece, usually, the theme recurs in the form of repetitions and variations. Barlow and Morgenstern compiled "A Dictionary of Musical Themes" that first appeared in 1948 [12]. In this book (referred to as BM in the following), the authors listed nearly ten thousand musical themes from instrumental musical works. Figure 5.1a shows a

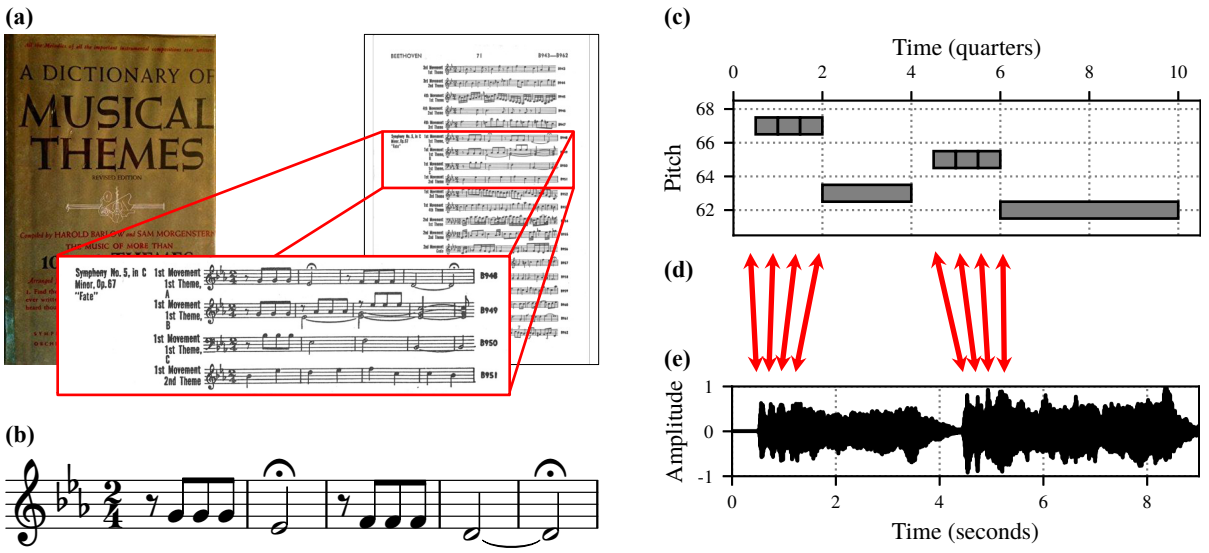


Figure 5.1: The BM book and various derived data modalities from MTD. **(a)** Original book. **(b)** Clean sheet music engraving of a musical theme. **(c)** Piano roll representation. **(d)** Alignment data. **(e)** Waveform of audio snippet.

page from the BM dictionary and a detailed view of the first theme of Beethoven's Symphony No. 5. John Erskine explains in the book's introduction that „the ten thousand themes [. . .] do not encompass the entire literature of music, but they do include practically all the themes which can be found in compositions that have been recorded." Though this statement from 1948 may not be valid today, it shows how ambitious the book was perceived at the time.

This chapter describes a multimodal dataset, called MTD (musical theme dataset), that is inspired by the original BM dictionary. Using a subset of 2067 themes of the BM book, we digitized and extensively augmented the material, and provide several digital representations of the musical themes. Beyond graphical sheet music (Figure 5.1b, similar to the BM dictionary), the dataset encompasses symbolic encodings (Figure 5.1c) of the themes in various formats (MIDI, MusicXML, CSV). As one main contribution of the MTD, we annotated the occurrences of the themes in audio recordings and provide snippets from these recordings corresponding to the annotated occurrences (Figure 5.1e). We also provide machine-readable metadata concerning the composer, work, recording, and musical characteristics. As another major component, we manually time-aligned the symbolic encodings to the audio snippets (Figure 5.1d). We provide alignment information as well as modified symbolic encodings that are synchronized to the audio versions. These links can be seen as note-level annotations of the audio material, yielding valuable fine-grained reference annotations for tasks such as audio transcription and cross-modal retrieval. A special feature of the dataset is that the themes are monophonic, while they usually appear in a polyphonic context in the recordings. The difference in the polyphony of the modalities allows for studying tasks related to melody extraction and source separation. All the modalities are easily accessible through our web-based interfaces. In addition, we provide basic tools for parsing, visualizing, converting, and processing the various data modalities. In this way, we bridge the gap between the printed BM book and

MIR research. We also provide our custom tools for manually aligning the symbolic and audio versions to enable the users of the MTD to continue expanding the dataset.[17]

We refer to the dataset as multimodal because it contains representations on various semantic levels (symbolic, audio, image). Other meanings of the term "modality" may instead refer to sensations like vision, touch, hearing, and kinematics [191]. In this chapter (and throughout this thesis), we do not use the term in the latter way.

The MTD is relevant to the MIR community in various ways. The correspondences between the different modalities (symbolic, audio, image) can be used for cross-modal retrieval [136]. Actually, preliminary versions of what have become the MTD already have been used for music retrieval experiments [10, 215, 221]. The manual alignments constitute valuable material for automatic music alignment [6, 88, 131]. The graphical sheet music can be used for OMR [33, 36, 156]. The dataset can also be useful for music transcription [16] or melody extraction [167]. Finally, aspects of polyphony could be interesting for the subfield of computational musicology [200].

This chapter is structured as follows. In Section 5.2, we discuss related datasets and summarize the MIR literature that is related to aspects of the BM book. Then, in Section 5.3, we address the role of musical themes in Western classical music and provide some examples from the BM book. As the main contribution of this chapter, we describe in Section 5.4 the MTD and its modalities, and then introduce in Section 5.5 our web-based interfaces and tools. Finally, to illustrate the potential of the MTD, we discuss in Section 5.6 possible applications and future work.

5.2 Related Datasets and Literature

5.2.1 Datasets

A diverse range of research datasets has been published by the MIR community. The first larger music dataset compiled specifically for research purposes is the RWC music database [69]. More recent examples are the multitrack dataset MedleyDB [22] or the Erkomaishvili dataset for ethnomusicological research [162].[18] Serra [179] discussed the role of datasets for the MIR community. Specifically, he distinguishes between rather unstructured datasets and curated research corpora. In particular, he introduces five criteria (purpose, coverage, completeness, quality, reusability) that are essential for a research corpus. It that sense, the MTD and many other datasets mentioned in this chapter, can be regarded as research corpora. In the following, we discuss some datasets that are more closely related to the MTD.

[17] We provide the raw data, an overview website, along with interfaces and tools on an accompanying website. `https://www.audiolabs-erlangen.de/resources/MIR/MTD`

[18] A list of datasets for MIR is to be found at `http://ismir.net/resources/datasets`.

Most of the BM themes have been available as symbolic versions (MIDI) at a website called *The Multimedia Library*, developed by Jacob and Diana Schwartz. Unfortunately, the page is now offline and the MIDI files have been withdrawn due to copyright reasons. Currently, the page is only reachable with the Wayback Machine without access to the MIDI files.[19] The dataset was denoted as the electronic dictionary of musical themes (EDM). While the EDM yields MIDI files for all ten thousand BM themes, the MTD provides a wealth of different representations and tools for two thousand of these themes.

There are related datasets containing main melody annotations, such as the Orchset [27] for orchestral music recordings, and MedleyDB [22] mainly for popular music recordings. Datasets with main melody annotations also exist for purely symbolic music [180]. Although a musical theme can occur as a main melody, the concepts are not identical. A main melody is a salient element in an excerpt of music and does not necessarily play an important role in the composition. In contrast to that, a theme is a musical idea that can occur in different ways within the musical context (see Section 5.3 for various examples). Furthermore, a theme is important throughout the musical piece because it typically recurs in the form of repetitions and variations in the course of the composition. Another distinction of the MTD is its diverse instrumentation, which is not restricted to orchestral music but also contains instrumental solo pieces and chamber music.

Several datasets for automatic music transcription (AMT) provide audio recordings of musical pieces and symbolic encodings that are synchronous with the recordings. In particular, many datasets focus on piano music. Examples are the MAPS database [57], the SMD [134], or the Maestro dataset [78]. For these AMT datasets, the alignments between the symbolic and audio representations are obtained by using hybrid acoustic/digital player pianos. In contrast to that, the MTD contains audio from commercial recordings that are manually aligned to the symbolic encodings. Another AMT dataset is MusicNet [190], which provides audio recordings and symbolic encodings of Western classical music pieces in diverse solo and chamber music instrumentations. For this dataset, the alignments between audio and symbolic representations have been created fully automatically using dynamic time warping. A further related dataset is MSMD containing MIDI representations, graphical sheet music, and synthesized audio for classical piano pieces with note-level alignments between the modalities [53]. Though rather designed for sheet music retrieval and score-following, it can also be used for AMT. Unlike the MTD, AMT datasets (such as MAPS, SMD, Maestro, MusicNet, and MSMD) contain all note events of the polyphonic pieces. In contrast, we provide the note events of the monophonic themes, even if they appear in a polyphonic context. While the mentioned datasets are better suited for polyphonic music transcription, the MTD is more appropriate, e.g., for melody estimation, or for studying questions concerning musical themes, saliency, and polyphony.

Other related datasets are used for the MIR task of query-by-humming (see Section 5.2.2 for more details). For example, the MTG-QBH dataset [166] contains monophonic recordings of melody excerpts from

[19] `https://web.archive.org/web/20160209045946/http://www.multimedialibrary.com/barlow/index.asp`

Table 5.1: Categorization of theme/melody retrieval scenarios, according to modalities.

Query	Database	Example Literature
symbolic	symbolic	[93]
audio	symbolic	[145]
symbolic	audio	[10]
audio	audio	[166]

amateur singers without symbolic representations. Given a set of polyphonic audio recordings where the same melodies occur, one can compare the a cappella recordings with the polyphonic recordings.

5.2.2 Literature

The BM dictionary contains a book index for finding the musical themes in the dictionary. To get the index term for a theme, one first transposes it to C (C major for major keys and C minor for minor keys). The first pitches from the theme without octave information are then used as an index term. This concept (pitches without octave information) is similar to the concept of pitch classes. However, for the index terms, one keeps enharmonic spellings, which is not always done for pitch classes. For example, let us consider the first theme of Beethoven's Fifth Symphony (see Figure 5.1), which does not need to be transposed because it already is in the key of C minor. Its index term in the book is (G, G, G, $E^{\flat}$, F, F). In the book, the index term is paired with a theme identifier, which allows the reader to quickly find the page of the theme.

This indexing scheme influenced several algorithmic approaches to musical search. There exist many different ways to realize an automated search engine for melodies and themes. One may categorize search scenarios according to the modalities used for the query and the database. Table 5.1 shows such a categorization, where a single reference for each category is provided as an example. An early example for symbolic–symbolic search is Themefinder[20], which provides a web-based interface for searching musical themes or incipits [93]. A more recent example is the online music catalog RISM[21], where a diatonic incipit search used to be offered, using index terms similar to the BM index, but without accidentals [46].

An undertaking similar to the BM book is the dictionary by Parsons [140]. In this book, compared to Barlow and Morgenstern [12], the indexing technique for musical melodies is further developed. The index term for a theme is here defined by its contour, where one specifies for each note if its pitch goes up (u), down (d), or repeats (r) compared to the previous note. The symbol * denotes the beginning of the sequence. For our Beethoven example the index term is (*, r, r, d, u, r, r, d). Researchers from the field of MIR investigated the benefits and limitations of the indexing schemes by Barlow and Morgenstern [20] and Parsons [196]. Prechelt and Typke [145] used the Parsons code in their query-by-humming system called Tuneserver. In this system, the user specifies a query by whistling or humming a melody. This query is then transcribed and compared with the symbolic melodies of the dictionary using the Parsons code. In this retrieval scenario, they used audio-based queries and a database of symbolic themes. There

[20] `http://www.themefinder.org`
[21] `http://www.rism.info`

is also a kind of opposite retrieval scenario, where the query consists of a symbolic encoding of a musical theme, which is then used to identify music recordings that contain this theme [10, 215, 221]. We will describe this scenario in further detail in Section 5.6 and in the next chapter. Other retrieval scenarios use audio representations for both queries and database documents [166].

The BM themes have also been used in the MIR community for classifying composers [144] and for segmenting themes in polyphonic symbolic music [121]. London [110] examined the BM book in a meta-study on building representative music corpora and considered it a useful collection. Of course, the BM dictionary also influenced and inspired research outside music retrieval research. Examples are a theoretical study about musical intervals [201], psychological music articles [181, 182], and a book about the origin and evolution of speech and music [43].

5.3 Musical Themes in Western Classical Music

As Drabkin [54] describes, a theme is "the musical material on which part or all of a work is based." Going even beyond that, Reti [158] describes the thematic process in a composition as its main form-building element, creating unity even across multiple movements of a work. The musical term "theme" originates from the 16th century. As we understand it today, a theme is a musical idea that conveys a sense of "completeness" and "roundedness," in contrast to the shorter and more basic musical motif. An essential aspect of Western classical music is the repetition and variation of thematic material throughout a musical work. In musicology, it is not well defined if a theme is only a monophonic melodic line, independent from the polyphonic context in which it occurs, or an entire polyphonic section [54]. However, in this thesis, we always consider a theme as being monophonic. Sometimes, it can be subjective whether a melodic line is a theme, and even musicologists may not agree upon this. In our context, we use the BM dictionary as reference to identify musical themes.

Musical themes can have different degrees of prominence or salience within their polyphonic contexts. In the following, we have a look at four different examples with a decreasing degree of salience. The first example is again the famous "Fate motif" from Beethoven's Fifth Symphony. Figure 5.2a shows a piano transcription of the section where the theme occurs. Although the name suggests that it is a motif, it is classified as a theme in the BM book. One might consider the first four notes as a motif, while the theme consists of two motif statements at different diatonic pitches. In this example, the theme is played by all instruments, in different octaves. As a consequence, only a single pitch class is present at a time. In the BM dictionary, the pitches of the upper staff constitute the theme (colored in red in Figure 5.2a). As a second example, we consider the second theme in the first movement of Beethoven's Piano Sonata Op. 2, No. 2 (see Figure 5.2b). Here, a main melody appears with a harmonic accompaniment, which is a typical situation for a musical theme. The sixteenth notes of the accompaniment present a minor triad (E, G, B) in the first half and a diminished triad ($F^\sharp$, A, C) in the second half. The theme is still prominent since it contains the highest pitches and is the only melodic line in this section. The third example is the

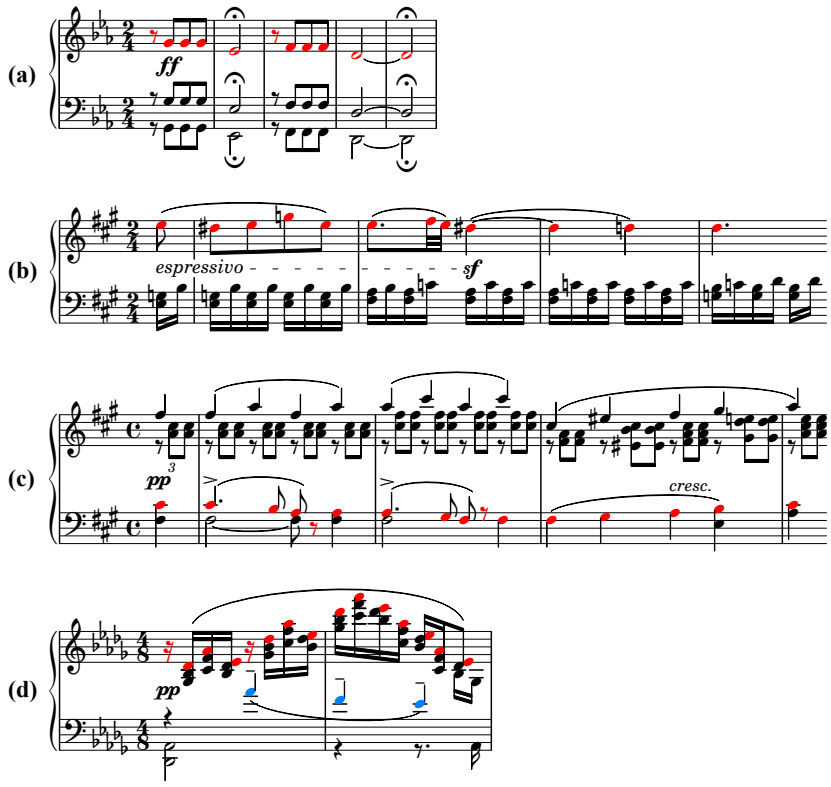


Figure 5.2: Various themes from the MTD. **(a)** Beethoven: Symphony No. 5 in C minor, Op. 67, first movement (piano transcription), first theme. **(b)** Beethoven: Piano Sonata No. 2 in A major, Op. 2, No. 2, first movement, second theme. **(c)** Schubert: Piano Sonata in B♭ major, D 960, first movement, second theme. **(d)** Debussy: *Reflets dans l'eau* (*Images*, Book 1, L 110, No. 1), two themes.

second theme in the first movement of Schubert's Piano Sonata D 960 and is shown in Figure 5.2c, where the red noteheads indicate the theme. This theme is less prominent because, first, it is in a middle voice and, second, the upper voice also is a melodic line, though with less independence. The fourth example (Figure 5.2d) is the beginning of the first piece of the suite *Images* by Debussy. This is a complex case because two different themes are overlapping. One theme is played by the right hand (upper staff, colored in red), and another short theme is played with the left hand (lower staff, colored in blue). In the BM book, both themes are referenced with a single identifier. However, for MTD, we gave two identifiers for the respective themes. Furthermore, the lower-staff theme again illustrates that there is no strict boundary between a musical theme and a motif. One may argue that this basic three-note sequence is to be regarded as a motif instead of a theme. However, Barlow and Morgenstern classified it as a theme in their dictionary.

5.4 Dataset

As our main contribution, in this section, we describe the MTD dataset. We start by discussing the origins of the MTD in the BM book and the EDM (Section 5.4.1). Next, we explain our collected metadata (Section 5.4.2). We then describe the various symbolic encodings of the MTD (Section 5.4.3) and the audio recordings (Section 5.4.4). The alignment between the symbolic and audio representations is then explained in Section 5.4.5. Finally, we summarize the directory and file structure of the MTD (Section 5.4.6).

5.4.1 BM and EDM

The BM book, which contains nearly ten thousand themes, is considered a good starting point for building a representative corpus of classical music [110]. The book lists the themes along with an identifier and basic information about the composer and the musical work. The electronic dictionary of musical themes (EDM, see Section 5.2.1) contains the BM themes as MIDI files, which are named by a one- to four-digit number. The enumeration does not follow the BM book's order, which makes it hard to link the EDM collection to the BM book.

The first step in processing the EDM files was to identify the corresponding themes in the BM book. We then re-enumerated the MIDI files strictly in the order of the BM book. This new number is used as an identifier for the MTD. Because our dataset contains not all themes, but a subset, the list of the MTD identifiers has gaps.

The MTD was initially designed as a test scenario for cross-modal retrieval experiments. Then the dataset was successively broadened by adding more modalities, metadata, and alignment data. This makes the dataset a valuable testbed for various MIR tasks. Since the preparation of the modalities is labor-intensive, we restricted ourselves to 2067 themes by well-known composers. When selecting the MTD themes, we were guided by several practical considerations rather than musical guidelines. We preferred sets of themes from the BM book that correspond to complete work cycles, e.g., Beethoven's complete piano sonatas. In particular, we considered musical works contained in comprehensive CD album collections (e.g., Brilliant Classics' "Complete Edition" of works by Beethoven), such that a single album collection covers many themes of the MTD (see also Section 5.4.4). Furthermore, we selected works from the standard repertoire, where many performances are easily available, such that users of the MTD can add further audio occurrences to the dataset. Due to these considerations, the MTD is not musically balanced in a stricter sense (e.g., in terms of periods or genres). Even though this imbalance may be problematic for musicological studies, the variety of themes in the MTD is useful for MIR applications (such as described in Section 5.6).

For the 2067 themes, we provide complete coverage of all modalities and metadata, described in the following subsections.

5.4.2 Metadata

In the MTD, we provide detailed metadata on the composer, work, recording, and musical characteristics of the themes, which are described in the following.

As a main contribution, we identified the catalog number for each musical work that contains a theme. Consistent catalog-based work information is not specified in the BM book. For example, for Beethoven's Fifth Symphony, the Opus number 67 is given in the BM book. But as another example, for all works by J. S. Bach, the BM book does not provide a catalog number, such as BWV 1046 for the first Brandenburg

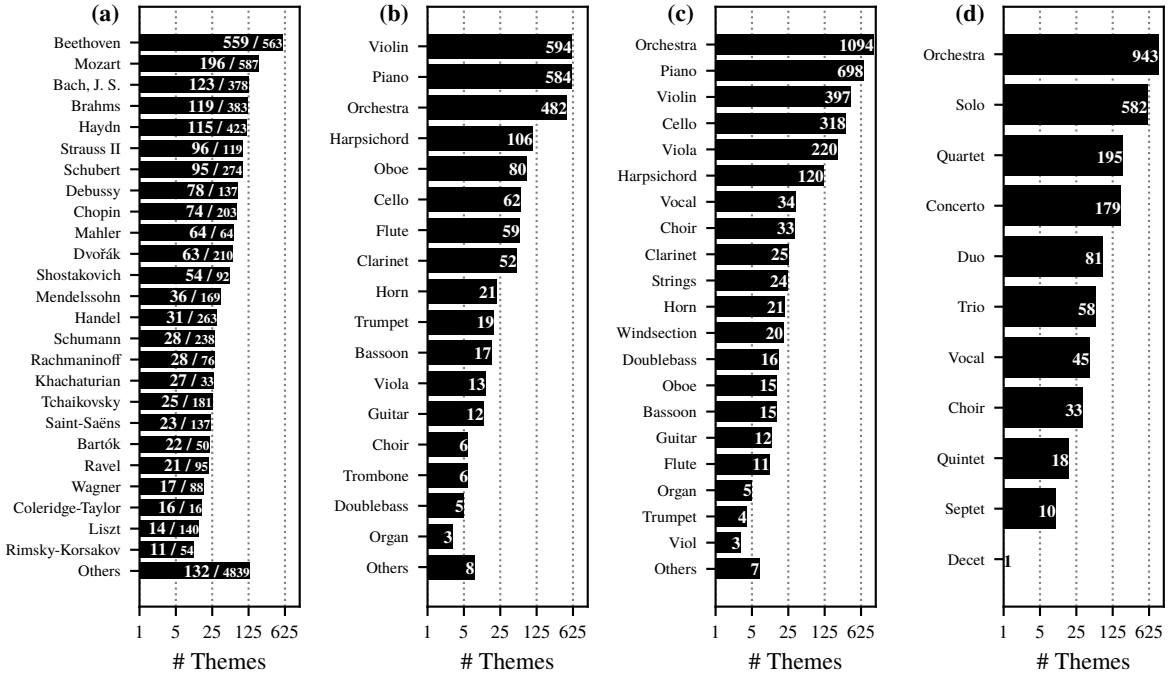


Figure 5.3: Various bar graphs for metadata of the themes. Numbers of themes on horizontal axes (logarithmic) are shown per **(a)** composer (smaller-font numbers after the slash indicate the total number of themes per composer in BM book), **(b)** theme instrumentation, **(c)** work instrumentation, and **(d)** ensemble type.

Concerto. In the case of the well-known BWV catalog of works by Bach from 1950, this was impossible because it was first published after the BM book from 1948. For our MTD work identifiers, we always use standard work catalogs when available. These identifiers relate to the movement level for multi-movement works. For example, the work identifier for the first movement in Bach's first Brandenburg Concerto is `BWV1046-01`.

Overall, we have 54 composers in our dataset. Figure 5.3a shows a bar graph of the number of themes per composer. In this figure, we only show composers with more than ten themes. We see that the most prominent composer of the dataset is Beethoven, with 559 themes. The second most common composer is Mozart, with 196 themes, followed by Bach, Brahms, and Haydn with a little more than 100 themes each.

Our metadata also contains several annotations regarding musical instrumentation. We show the distribution of these annotations in the Figures 5.3b–d. The bar graphs show the number of themes per instrumentation of the theme melody, per instrumentation of the entire work, and per ensemble type. The BM book specifies the instrumentation only to a certain degree. For example, a keyboard work by J. S. Bach could be played by a piano or by a harpsichord. Figure 5.3b shows the instrumentation of the themes as they occur in our selected recordings (the recordings are explained in Section 5.4.4). Note that a theme can be played by more than one instrument. That is why the overall bar graph count amounts to more than 2067 themes. We see that the dominating theme instruments are violin, piano, and orchestral tutti. But there are also several themes played by other instruments, such as harpsichord, oboe, cello, flute, or clarinet. It may

Field	Description
MTDID	Identifier, used in the MTD
BMID	Identifier, from original BM book
EDMID	Identifier, used in the EDM
ComposerID	Identifier, based on composer's name
WorkID	Identifier, usually based on catalog number
PerformanceID	Identifier, based on main performer of recording
CollectionID	Identifier, based on album collection
LabelID	Identifier, based on recording label
WCMID	Internal ID for audio recording
MusicBrainzID	MusicBrainz release ID for album collection
ComposerBirth	Composer's year of birth
ComposerDeath	Composer's year of death
WorkTitle	Sub-title, nickname, or non-numeric title for musical work
ThemeLabelBM	Label for theme, from original BM book
ThemeInstruments	Instrument(s) playing the theme
WorkInstruments	Instrument(s) of the musical work
Ensemble	Ensemble type
Polyphony	Indication of musical texture
NameCD	CD name in album collection
NameTrack	Track name in the CD of the album collection
StartTime	Start time of theme occurence in audio recording
EndTime	End time of theme occurence in audio recording
MidiTransposition	Pitch transposition difference between recording and symbolic encoding
Comment	Textual comment

Table 5.2: Overview of all metadata contained in the MTD.

be surprising that the choir appears as a category because the BM book only covers instrumental musical works. After the original BM book, Barlow and Morgenstern also published a separate dictionary of vocal music [13], which we did not use for the MTD. However, in the BM book there are a few exceptions, such as the choral finale of Beethoven's Ninth Symphony.

Figure 5.3c shows a bar graph of the instrumentation of the musical works. This instrumentation now refers to the entire piece of music and not just to the instruments which play the themes. For example, the third theme in Beethoven's Fifth Symphony is played by the horn, but the instrumentation of the corresponding work is the entire orchestra. Finally, Figure 5.3d shows a bar graph of the ensemble type. For nearly half of the themes (943) the ensemble is the full orchestra, and more than a quarter (582) of them are from solo pieces. Another quarter of the themes are played by other types of ensembles, such as quartets or duos.

As a further musical characteristic, we also provide annotations for the musical texture of the music segments where the themes appear. Studying texture is a challenging topic on its own [67]. Following the textbook by Benward and Saker [19], we annotate the texture of the themes according to the standard categories of monophony, homophony, and polyphony. A monophonic texture consists of a single melodic

line (possibly doubled by octaves). A homophonic texture is made up of a melody and an accompaniment. A polyphonic texture comprises two or more independent melodic lines. According to Benward and Saker [19], there is also the fourth category of homorhythmic texture with a similar rhythm in all voices. However, for our annotations, we include cases of this category into homophony. Of course, more than one texture can appear in a single theme. For example, the beginning of a fugue often starts by presenting a musical theme (called the subject in the case of fugues) in a monophonic way. Still, the texture turns polyphonic as soon as another voice joins in (which can be before the end of the subject). In such cases, we assign multiple categories. Even though our coarse categorizations have to be taken with care, they may serve, e.g., as a guiding principle when evaluating MIR applications using the MTD. One may decide on a different category for border cases, but the annotations should be appropriate for unambiguous cases.

Table 5.2 shows all metadata fields of the MTD with a short description of each. The upper part of the table shows various identifiers, and the lower part descriptive metadata fields. Some of the entries relate to the audio recordings, which are described in Section 5.4.4.

5.4.3 Symbolic Encodings

We provide various symbolic encodings of the musical themes. As one main contribution, we newly engraved the themes with the sheet music editor Sibelius. We denote this encoding by SCORE. Beyond the Sibelius files, our dataset contains exports to PDF, MIDI, MusicXML, and CSV. Among these formats, the only non-standard music format is our CSV representation, which encodes each note's start, duration, and pitch in a simple way.

While building the MTD dataset, we worked with the original EDM files parallel to engraving the SCORE versions. For this reason, we also provide the original EDM files of the MTD themes (denoted by EDM-orig). The EDM files often contain errors, e.g., wrong pitches and rhythms, or missing and incorrect ornaments. As one of our contributions, we consistently corrected all wrong pitches in the MIDI files. We also fixed some rhythm and ornament errors without being comprehensive here. We provide our corrected EDM files (denoted by EDM-corr) additionally to the original ones.

In some cases, we also found errors in the original engraving of the themes, where the BM book unintentionally deviates from the corresponding musical work's score. In these cases, we corrected the errors (for SCORE and EDM-corr) to be consistent with the score.[22] A general principle of the MTD is that a theme is always a monophonic note sequence. As for the BM book, however, there are few exceptions, where the theme is notated in a polyphonic way. For these examples, we decided on a monophonic representation of the theme (for EDM-corr).[23]

[22] Out of 2067, this affects 30 themes, namely the ones with the MTD IDs 0770, 1033, 1109, 1143, 1484, 1501, 1737, 1742, 1788, 2609, 2619, 2966, 3944, 4287, 4305, 5323, 5566, 5753, 6008, 6840, 7636, 7670, 8111, 8137, 8141, 8355, 8549, 8560, 9130, and 9516-2.

[23] Out of 2067, this affects 18 themes, namely the ones with the MTD IDs 0261, 0389, 0411, 0429, 0433, 0435, 0765, 0957, 1035, 1067, 1162, 2583, 2597, 6803, 7232, 7718, 8470, and 8780.

(a)

# Themes	2067
# Composers	54
Symbolic	
Σ Dur. (quarters)	29 359.8
⌀ Dur. (quarters)	14.2 ± 6.0
Audio	
Σ Dur. (hh:mm:ss)	4:57:48.3
⌀ Dur. (seconds)	8.6 ± 5.6

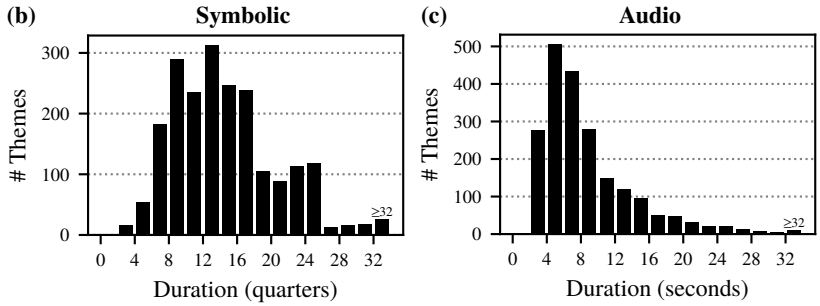


Figure 5.4: Dataset overview. **(a)** Main statistics of dataset. Average information (⌀) given as mean ± standard deviation. **(b)** Histogram of theme durations in quarter notes and **(c)** in seconds.

We also always assume that a theme is a continuous sequence of notes and rests. In almost all cases, this assumption is fulfilled in the BM dictionary. However, for a few instances in the BM book, a single identifier is used to denote two themes. In these cases, the themes are either separated by a gap of several measures or overlapping in time (we discussed such a situation in Section 5.3, Figure 5.2d). For these exceptions, we deviate from the BM book and give individual identifiers for the themes.[24]

Both EDM-orig and EDM-corr are symbolic representations that do not focus on the sheet music layout. In contrast, our new SCORE engravings are created with Sibelius and can be used for graphical purposes.

We now discuss some statistical aspects of the symbolic encodings of the MTD themes. Figure 5.4a shows some general statistics of the dataset. The average duration of the 2067 themes is 14.2 quarter notes. The entries for the audio representations will be explained in Section 5.4.4. Figure 5.4b shows a histogram of the themes' durations (based on EDM-corr), measured in quarter notes. Most themes have a length of 6 to 18 quarter notes. However, a few themes have a short duration of below four quarter notes or a long duration of more than 30 quarter notes. Of course, in different time signatures, quarter notes have different meanings. In the bar graph of Figure 5.5, we show the number of themes for different time signatures. In the case of multiple time signatures for a single theme, we only use the first one for this figure.[25]

5.4.4 Audio Recordings

For each musical work of the MTD themes, we selected a recording contained in a comprehensive CD album collection, where a single collection typically covers many themes in the MTD. Overall, the 2067 occurrences are to be found in 61 album collections. To identify these collections, we provide MusicBrainz IDs [189] in our metadata. The full list of MusicBrainz IDs is also available as a MusicBrainz user collection.[26]

[24] This is the case for 9 BM IDs, which we split into 18 MTD IDs, namely 0739-1/2, 0755-1/2, 0768-1/2, 2255-1/2, 2332-1/2, 2338-1/2, 2631-1/2, 9164-1/2, and 9516-1/2.

[25] Out of 2067 themes, 16 have multiple time signatures, namely the ones with the MTD IDs 0579, 0585, 3622, 5332, 5337, 5354, 5364, 6777, 7230, 8052, 8131, 8147, 8150, 8782, 8794, and 8795.

[26] `https://musicbrainz.org/collection/57faf42e-045e-415d-9f1d-dd5d0699b689`

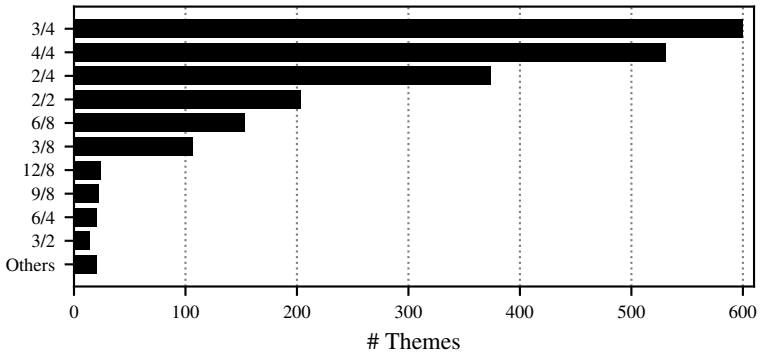


Figure 5.5: Time signatures in the MTD.

For each theme, we decided on one prominent occurrence in the recording. Typically this is the first occurrence of the theme. Then, we annotated the beginning and end of the theme occurrence in the recording. We generated audio excerpts corresponding to these occurrences, which are a central component of MTD. Since these excerpts are concise, we consider providing the audio files as fair use. As shown in the table of Figure 5.4a, the occurrences have an average duration of 8.6 seconds. Figure 5.4c shows a histogram of the durations of the audio snippets. This distribution is strongly skewed, with most themes having a duration between 4 and 6 seconds.

We observed that several theme occurrences are transposed compared to the versions in the BM book. The reason for this may be that the entire recording is transposed or that the reference for the BM book is a different occurrence of the theme. We consistently annotated the transposition differences in semitones.

5.4.5 Alignment Data

As a further main contribution, we manually aligned the symbolic themes (EDM-corr) to the audio recordings. To support this process, we created a custom web interface for editing the alignment and listening to the result in the form of a superposition of the audio snippet and a synthesized version of the aligned theme (more details in Section 5.5). This sonification helped us to check the edited alignment. We created alignment paths that consist of pairs of corresponding time points in the audio and the symbolic representations, respectively. They enabled us to synchronize the symbolic representations (EDM-corr) to the audio recordings. We additionally provide these modified symbolic representations (denoted by EDM-alig).

5.4.6 Directory and File Structure

The structure of the MTD with regard to its modalities yields a natural directory structure. There are several directories on the top level, each containing a different data modality for all themes. Table 5.3 lists the directories of the MTD. All files inside the directories are consistently named using the MTD ID, composer, and work identifiers, e.g., MTD1066_Beethoven_Op067-01.

Directory	Description	Format
data_EDM-orig_CSV	Original EDM files	CSV
data_EDM-orig_IMG		PDF
data_EDM-orig_MID		MIDI
data_EDM-corr_CSV	Corrected EDM files	CSV
data_EDM-corr_IMG		PDF
data_EDM-corr_MID		MIDI
data_EDM-alig_CSV	Aligned EDM files	CSV
data_EDM-alig_MID		MIDI
data_SCORE_CSV	SCORE files	CSV
data_SCORE_IMG		PDF
data_SCORE_MID		MIDI
data_SCORE_SIB		Sibelius
data_SCORE_XML		MusicXML
data_ALIGNMENT	Alignment data	CSV
data_AUDIO	Audio snippets	WAV
data_META	Metadata	JSON

Table 5.3: Overview of the MTD directory structure.

5.5 Interfaces and Tools

We provide access to the MTD in three different ways.[17] The first way is to download an archive with the raw data (see also Table 5.3). The second way is a website that presents the different data modalities of the dataset. Third, we provide a Jupyter notebook containing Python code for parsing, visualizing, and sonifying the data. In this section, we introduce the website and the Jupyter notebook. Furthermore, we also describe our custom tool for aligning the symbolic and audio representations.

On the website's start page, we list all 2067 themes of the MTD in a table, see Figure 5.6a. For each theme, there is a dedicated subpage. The subpages can be accessed by clicking on the MTD ID in the table. For example, in Figure 5.6a, we highlighted the link for the MTD ID 1066. Figure 5.6b shows a screenshot of the subpage of this theme. On the top, we display three variants of graphical sheet music (EDM-orig, EDM-corr, and SCORE) along with corresponding MIDI playback buttons.[27] Note that the images for EDM-orig and EDM-corr are generated from MIDI (using the software MuseScore), which is not meant for graphical sheet music rendering. Even though the corresponding piano roll representations are accurate, the graphical rendition may not be musically meaningful. However, the images of the SCORE versions are of consistent quality because we directly engraved the sheet music using Sibelius. Furthermore, we offer two versions of the theme's occurrence in an audio recording: the first version is the respective audio excerpt, and the second one is a mixture of the same excerpt and a sonification of the aligned theme (EDM-alig). Finally, we show a table with the metadata.

[27] The MIDI playback is realized with the Javascript libraries MidiPlayerJS (`https://github.com/grimmdude/MidiPlayerJS`) and soundfont-player (`https://github.com/danigb/soundfont-player`).

Figure 5.6: Screenshots of our web-based interfaces. **(a)** Overview table of web page. **(b)** Subpage for the theme with MTD ID 1066. **(c)** Jupyter notebook. **(d)** Alignment tool.

The website is an easy way to explore the MTD, but it is static. In contrast to that, our Jupyter notebook allows for interaction with the data. We build upon standard Python packages such as pretty_midi [151] and librosa [119], and use the Jupyter framework, which is common in the MIR community [130]. Figure 5.6c shows a screenshot of the notebook. In this part of the notebook, we first load the audio snippet for the first theme of Beethoven's Fifth Symphony. Then, we compute a spectral representation with logarithmic frequency spacing [171]. We visualize the spectral matrix as a grayscale image. Because the frequency bandwidth corresponds to a semitone, the frequency axis can also be used as the pitch axis of a piano roll representation. Using the aligned symbolic music encoding (EDM-alig), we can superimpose a piano roll visualization (red color) on the image. Thus, we highlight the spectral bins that coincide with the fundamental frequencies of the theme's notes.

We also provide our custom alignment tool for manually creating the alignments between the audio snippets and the MIDI files. This web-based tool is not a general-purpose product, but its source code

may be useful for the MTD users who want to further refine the alignments or add further themes to the dataset. The tool is a client-server application that uses the Python web development package Flask[28] for its back end and the Javascript canvas library Fabric.js[29] for the graphical elements in the web front end. Figure 5.6d shows a screenshot of this web-based tool, which visualizes chromagrams for the audio snippet (upper) and the MIDI file (lower). In between, there are red lines that are connected to the chromagrams by red dots. These red dots indicate the time positions of the theme's note onsets in the representations of the respective chromagrams. In this way, the connecting lines define an alignment path between both representations.

The red dots are fixed in the MIDI version and flexible in the audio version. By moving the red dots in the audio version, the user can specify the onsets' time positions and, therefore, change the alignment. The alignment for all time points not related to note onsets is then obtained by linear interpolation. After clicking on the button for processing, a sonification is automatically generated that helps to evaluate the overall alignment accuracy. Finally, the alignment can be saved as a CSV file (same format as data_ALIGNMENT in the MTD). We also provide further Python scripts that use this CSV file to generate the time-aligned symbolic formats (data_EDM-alig_CSV and data_EDM-alig_MID).

There are many tasks where temporal annotations of high resolution are beneficial. As future work, our alignment tool may be modified to be useful for other applications, such as bird song recognition [125] and audio event detection [64].

5.6 Applications and Future Work

Due to its multimodal nature, the MTD can be useful for many MIR tasks. In this section, we discuss some possible applications and indicate future work directions.

Being inspired by the BM dictionary, our dataset offers links between the sheet music in the printed book and new digital engravings. This is an interesting testbed for OMR [33, 36, 156]. On the one hand, the monophonic themes constitute a relatively simple OMR scenario. On the other hand, the difference between modern versions and old engravings from the 1940s can be challenging. Balke et al. [9] already presented a study using OMR for the printed BM book. They report on retrieval experiments, where they aimed at finding relevant MIDI files in the EDM collection. The queries were generated using OMR and OCR processing of the BM book.

The new engravings of the MTD go along with symbolic representations. The correspondences between these symbolic representations and the audio occurrences open up possibilities for exciting cross-modal retrieval scenarios. For example, in one such task, using a theme's symbolic encoding as a query, the aim is to identify all relevant audio recordings that contain an occurrence of the query theme in an audio

[28] `https://flask.palletsprojects.com`
[29] `http://fabricjs.com`

database. In this retrieval scenario, main challenges are the differences in modality (symbolic vs. audio) and musical characteristics (monophonic vs. polyphonic). Several studies approached this task, and some already used preliminary versions of what have become the MTD [10, 215, 221]. The next chapters (Chapter 6 and Chapter 7) cover related retrieval approaches based on enhanced chroma features and local alignment techniques.

Another task that also involves symbolic and audio representations is to estimate the fundamental frequency (F_0) contours of the themes in the audio occurrences. This task is closely related to melody estimation [167]. A typical approach for this task is to first compute a pitch salience function, which is a time–frequency representation where the melody's F_0 components are enhanced, and other components are attenuated. In the second step, F_0 contours are tracked in the salience representation. Bittner et al. [23] proposed a machine-learning strategy to compute a salience representation using a CNN. Using such techniques, one may learn a salience function for musical themes employing the audio occurrences and the aligned symbolic representations of the MTD as a training set.

In even more challenging scenarios, one may aim to find direct correspondences between the audio recordings and the sheet music images, without utilizing symbolic representations. If the alignment of these modalities is performed online, the application is also known as score following. There are first works to learn representations for score following with data-driven approaches in an end-to-end fashion [51]. The correspondences between sheet music images and audio excerpts of the MTD can be used as additional training and test data for such approaches. A further challenge here is the monophonic–polyphonic discrepancy between the score–audio pairs.

A general problem in data-driven approaches is the need for aligned training data. The correspondences between audio excerpts and symbolic encodings in the MTD may serve different purposes since they are both weakly aligned (theme level) and strongly aligned (note level). For example, the MTD may serve as a testbed to develop and evaluate alignment approaches within deep-learning frameworks. An example of such an approach is the CTC loss [71], which can be used to train a neural network with weakly aligned data. Stoller et al. [185] used this loss to align music recordings to textual lyrics, where they only used weakly aligned audio–lyrics pairs for training. In another study, the CTC loss was used for OMR of monophonic music [34]. In Chapter 7, we use the weak and strong alignments of the MTD to develop and test a CTC-based learning approach within a challenging musical scenario.

As another application, the MTD alignments between the symbolic scores and audio recordings can be used for detailed analyses of the music performances [102], in particular in terms of tempo and timing [47]. For example, MIR researchers used alignment information to compute tempo curves that visualize the tempo change throughout the performance of a musical piece [133]. A web-based tool for tempo comparison was created by Peachnote.[30] Using the MTD, one may analyze the tempo characteristics of performances of musical themes.

[30] `https://www.tuttitempi.com`

Given the rich metadata of the MTD, the dataset may as well be valuable for various music recommendation and classification tasks. Examples are composer classification [198] and instrument identification [58].

In summary, the MTD offers a rich and diverse cross-modal dataset for music processing. The dataset may trigger future research on exploring the potential of musical themes and multimodality for MIR research.

6 Evaluating Salience Representations for Cross-Modal Music Retrieval

This chapter is based on [221]. The first author Frank Zalkow is the main contributor to this article. In collaboration with his supervisor Meinard Müller and his colleague Stefan Balke, he developed the ideas and wrote the paper. Building upon the previous work of Stefan Balke [10], Frank Zalkow implemented the approaches and conducted the experiments.

In this chapter, we consider a cross-modal retrieval scenario of Western classical music. Given a short monophonic musical theme in symbolic notation as a query, the objective is to find relevant audio recordings in a database. A major challenge of this retrieval task is the possible difference in the degree of polyphony between the monophonic query and the music recordings. Previous studies for popular music addressed this issue by performing the cross-modal comparison based on predominant melodies extracted from the recordings. For Western classical music, however, this approach is problematic since the underlying assumption of a single predominant melody is often violated. Instead of extracting the melody explicitly, another strategy is to perform the cross-modal comparison directly on the basis of melody-enhanced salience representations. As the main contribution of this chapter, we adapt and evaluate several conceptually different salience representations for our cross-modal retrieval scenario. Our extensive experimental results, which have been made available on a website, comprise more than 2000 musical themes and 100 hours of audio recordings.

6.1 Introduction

Ongoing digitization efforts create large amounts of music data in different modalities, such as audio recordings, symbolic representations, or graphical sheet music. Accessing this data in a convenient way requires flexible retrieval strategies that are able to cope with the different modalities. In the last decades, many systems for audio retrieval based on the query-by-example paradigm have been suggested. Given a fragment of a symbolic or acoustic music representation as a query, the task is to automatically retrieve documents from a music database containing parts or aspects that are similar to the query [40, 73, 152, 194]. One such retrieval scenario is known as *query-by-humming* [164, 166], where the user specifies a query by

singing or humming a part of a melody. The objective is to identify all audio recordings (or other music representations) that contain a melody similar to the specified query. In related retrieval scenarios, a short symbolic query is given, e.g., taken from a musical score, and the task is to identify another symbolic music representation [101, 112] or an audio recording [10, 60, 81, 143, 188].

Many pieces from Western classical music contain short melodies or musical gestures that are especially prominent and memorable (e.g., the famous "Fate Motif" at the beginning of Beethoven's Symphony No. 5). Finding such *musical themes* in audio recordings with computational methods constitutes a challenging retrieval scenario. In particular, given a symbolic representation of a musical theme as a query, the retrieval task is to find all recordings within a database of classical music recordings that contain this theme. Major challenges are due to the differences in modality (symbolic vs. audio), tuning, transposition, tempo, and degree of polyphony between the query and the database documents [10].

In this chapter, we built upon the results presented in [10], where the database and the queries are compared on the basis of chroma features. We take this work as a baseline for our follow-up study. In particular, we address a major issue of the previous work, which is the compensation of the difference in the degree of polyphony between the queries and the database documents. Instead of deriving chroma features from the full spectral content, we consider in this chapter several kinds of enhanced time–frequency representations, so-called *salience representations*, which emphasize certain tonal frequency components [128, Chapter 8] and enhance melodic structures in the spectrogram [11, 23, 26, 55, 165]. Previous studies [166] first extracted the predominant melody and bass line of the audio on the basis of salience representations. Then these representations are mapped to chroma features for performing the retrieval. However, in Western classical music, the underlying assumption of a single predominant melody is often violated, which has a negative impact on the robustness of melody extraction algorithms [167]. We propose not to extract the melodies but to map the salience representations directly to chroma features for retrieval.

The main contribution of this chapter is to evaluate several state-of-the-art salience representations—originally designed for melody extraction—for the given retrieval scenario, and conduct an extensive quantitative study exploring the potential of these representations. In Section 6.2, we describe the dataset used for the experiments. The feature representations used throughout this study are introduced in Section 6.3. In Section 6.4, we describe the retrieval procedure, report on results, and discuss the effects of the representations on the retrieval results by means of a new evaluation metric, called *separation indicator*. The results of our experiments have been made publicly available on an interactive website.[31] Finally, Section 6.5 presents a short conclusion.

[31] `https://www.audiolabs-erlangen.de/resources/MIR/2019-ICASSP-BarlowMorgenstern`

Table 6.1: Overview of the dataset. Duration format: hh:mm:ss.

Queries			Database			Composers
#	Mean Dur.	Total Dur.	#	Mean Dur.	Total Dur.	#
2045	00:00:09	05:00:03	1114	00:06:25	119:15:19	52

6.2 Dataset

The dataset considered in this study is a preliminary version of the MTD (introduced in Chapter 5). For convenience, we shortly describe the content of the dataset used in the experiments of this chapter.

As shown in Table 6.1, the dataset consists of 2045 themes (instead of 2067 themes as in the final MTD). A preliminary version of the MTD was also used in previous work [10]. Compared to this prior study, we substantially extended the dataset by annotating the occurrences of these themes in an audio collection, including the durations and possible transpositions. We designed the audio database in such a way that there is precisely one relevant music recording in the database for each query theme. Note that there can be more queries for a given audio recording: e.g., for the first movement of Beethoven's Symphony No. 5, there are six themes. The newly annotated audio material enables us to perform large-scale retrieval experiments in a controlled and systematic fashion, focusing on the monophonic–polyphonic matching problem. All queries of the dataset can be accessed through the accompanying website.[31] For more details on the MTD, we refer to Chapter 5.

6.3 Feature Representations

In this chapter, we consider various time–frequency representations that emphasize specific tonal frequency components. The considered approaches, which are well known in the literature, are listed in Table 6.2, with links pointing to implementations. Details and properties of the time–frequency representations are discussed in the next paragraphs. We convert these representations to time–chroma representations by suitably mapping the frequency bins to the twelve chromatic pitch classes, see [14, 68, 128]. In our study, we use a consistent feature rate of 10 Hz (applying median aggregation for representations with a higher feature rate). Furthermore, all frames of the chroma features are ℓ^2-normalized. Figure 6.1 visualizes the time–frequency representations (left column) and their derived chroma features (right column) for a music example.

In the case of a MIDI query, the feature extraction is straight-forward, see Figure 6.1a. While using a single feature representation for the MIDI query, we compare several chroma variants for the audio recordings. As a baseline, we use a time–frequency representation $\mathcal{S}_{\mathrm{IIR}}$ similar to a spectrogram with a logarithmically spaced frequency axis by using a bank of elliptic IIR filters (originally introduced in [127, Chapter 3], see also Section 2.4). This representation was also used for the experiments in [10]. Obviously, this approach is influenced by the complete spectral content present in the audio, such as harmonics or

	Ref.	Source for Implementation
$\mathcal{S}_{\text{IIR}}$	[127, 129]	`http://github.com/librosa/librosa` [119]
$\mathcal{S}_{\text{MEL}}$	[165]	`http://www.upf.edu/web/mtg/melodia`
$\mathcal{S}_{\text{SFM}}$	[55]	`http://github.com/juanjobosch/SourceFilterContoursMelody`
$\mathcal{S}_{\text{BG1}}$	[26]	
$\mathcal{S}_{\text{DNN1}}$	[11]	`http://www.audiolabs-erlangen.de/resources/MIR/2017-ICASSP-SoloVoiceEnhancement`
$\mathcal{S}_{\text{DNN2}}$		
$\mathcal{S}_{\text{CNN}}$	[23]	`http://github.com/rabitt/ismir2017-deepsalience`

Table 6.2: Table of implementations, used for computing the salience representations.

noise-like signal components. For instance, the beginning of C_{IIR} in Figure 6.1b has strong energy in the G-band, which corresponds to the fundamental frequency of the first note, but also in the D-band and in the B-band, which correspond to the third and fifth harmonics, respectively. A well-known approach that puts more emphasis on the predominant melody's fundamental frequency is harmonic summation, which is, e.g., used in MELODIA [165]. As shown in Figure 6.1c (first column), harmonic summation enhances the predominant melody in $\mathcal{S}_{\text{MEL}}$, but also introduces additional noise. Note that this representation (as well as many of the other salience representations) was designed to serve as input to a subsequent melody extraction step. Further traditional signal-processing approaches include a source-filter signal model $\mathcal{S}_{\text{SFM}}$, introduced by Durrieu et al. [55], and a combination of MELODIA with Durrieu's source-filter model $\mathcal{S}_{\text{BG1}}$, proposed by Bosch and Gómez [26]. For extracting $\mathcal{S}_{\text{BG1}}$, we use a threshold parameter setting (denoted by "BG1"), which turned out to be specifically suited for orchestral music [25].

Recently, deep learning became ubiquitous for computing salience representations. For instance, Balke et al. [11] used a fully connected neural network that was trained on jazz music for computing a salience representation specifically tailored to this kind of music, denoted as $\mathcal{S}_{\text{DNN1}}$. Since deep learning is highly data-adaptive and the network was not trained on Western classical music at all, we re-trained the model, following the training procedure as described in [11], using the publicly available Orchset dataset [27]. We applied pitch shifting as well as time scale modification to generate several versions of the dataset in different tempi and keys. As a result of this augmentation process, the data used for training was increased from 23.5 minutes to 820.5 minutes. The resulting salience representation is denoted as $\mathcal{S}_{\text{DNN2}}$. A more advanced deep-learning approach for computing a salience representation $\mathcal{S}_{\text{CNN}}$ was introduced by Bittner et al. [23]. They proposed a CNN architecture that uses a custom feature representation as input, called HCQT (see Section 2.5). The network was trained on classical, popular, as well as jazz music and outperformed state-of-the-art approaches in melody extraction.

Most salience representations were designed for a subsequent melody extraction step. We also perform this step for the three representations $\mathcal{S}_{\text{MEL}}$, $\mathcal{S}_{\text{BG1}}$, and $\mathcal{S}_{\text{CNN}}$, with the respective methods proposed by the original literature [23, 26, 165]. We consider all frames as voiced since this is the case for most queries.[32]

[32] Additional experiments (not reported here) showed that automatic voicing estimation leads to a drastic drop in retrieval quality in all three cases.

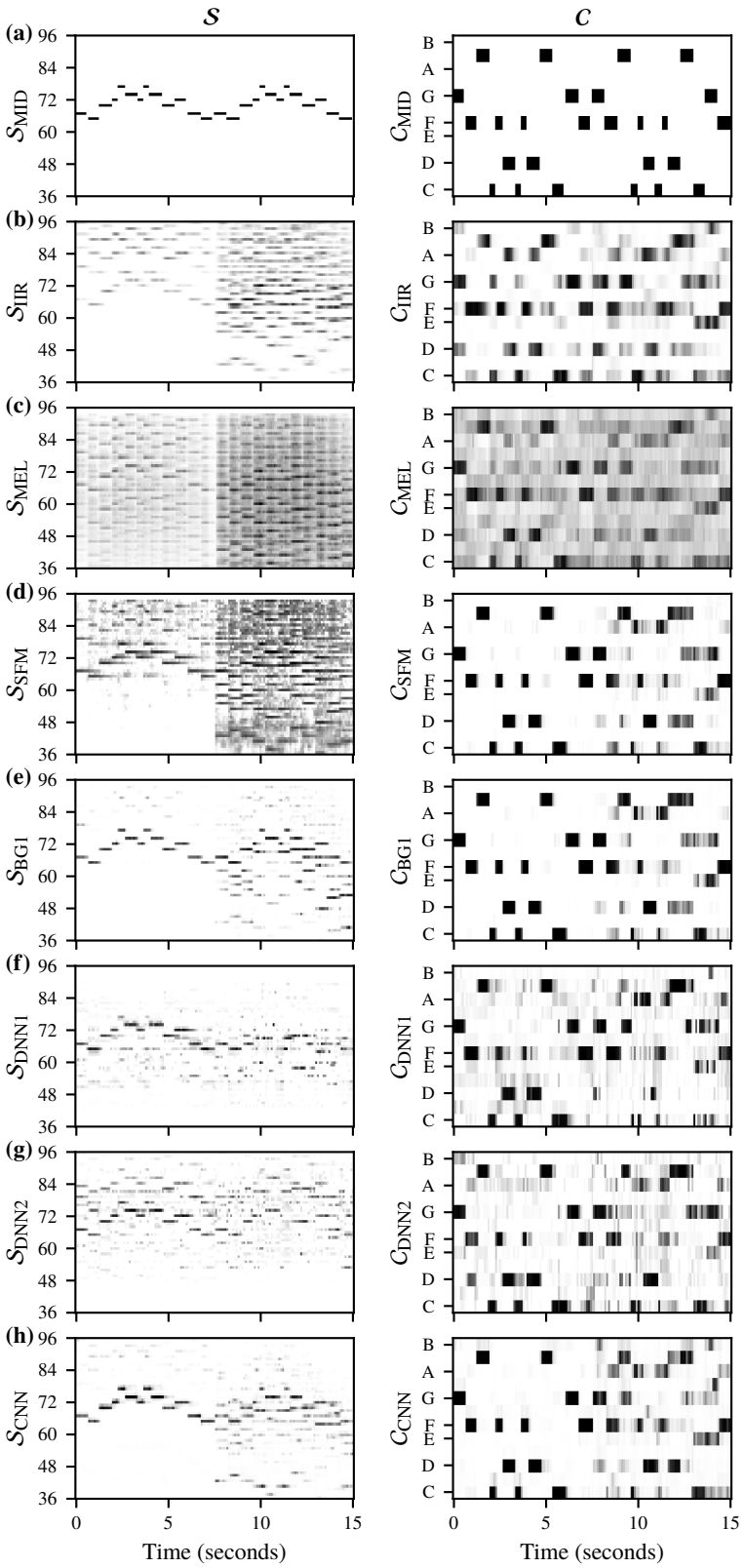


Figure 6.1: Overview of different time–frequency representations and their derived chroma features for the beginning of the first *Promenade* of Modest Mussorgsky's *Pictures at an Exhibition*. The first half of this example presents monophonic melody and the second half repeats it along with a homophonic brass section.

The extracted melodies are mapped to chroma features, just like for the time–frequency representations, resulting in C^*_{MEL}, C^*_{BG1} and C^*_{CNN}. As we will see in our experiments in Section 6.4.2, the melody extraction step is not beneficial within our retrieval scenario.

6.4 Experiments

In this section, we first summarize our retrieval procedure and describe our experiments. We then study how the different feature representations from Section 6.3 can cope with the difference in the degree of polyphony between the monophonic symbolic themes (queries) and polyphonic music recordings (database documents). Finally, we evaluate the approaches in depth.

6.4.1 Retrieval Procedure

We formalize our retrieval task following Balke et al. [10]. Similar procedures for synchronizing sheet music and audio recordings were described in the literature [60, 128, 188]. We use a collection of musical themes, where each theme Q is regarded as a *query*. Furthermore, let $\mathbb{D}$ be a set of audio recordings, which we regard as a database collection consisting of *documents* $\mathrm{D} \in \mathbb{D}$. Given a query, the retrieval task is to identify the semantically corresponding documents. Note that in our experimental setting, there is only a single relevant document for each query. To compare a symbolic query Q to a database document $\mathrm{D} \in \mathbb{D}$, we convert the query and the document into chroma sequences. Then, we use a standard technique known as subsequence dynamic time warping (SDTW) to compare the query with subsequences of the document, see [128, Chapter 7]. In particular, we use the cosine distance (for comparing ℓ^2-normalized chroma feature vectors), the step size condition $\Sigma := \{(2,1),(1,2),(1,1)\}$, as well as the weights $w_{\mathrm{vertical}} = 2$ and $w_{\mathrm{horizontal}} = w_{\mathrm{diagonal}} = 1$ in the SDTW.

As the result of SDTW, one obtains a matching function $\Delta^{\mathrm{Q}}_{\mathrm{D}}$ for a query Q and document D. Local minima of this function point to locations with a good match between the query Q and a subsequence of the document D. For a given query Q, the retrieval task can be solved by computing matching curves for all documents D, and by taking the minimum $\delta^{\mathrm{Q}}_{\mathrm{D}} \in \mathbb{R}_{\geq 0}$ for each of the matching functions $\Delta^{\mathrm{Q}}_{\mathrm{D}}$. The values of these minima yield a ranking of the database documents, which can then be presented in the form of an ordered list. The position of a document $\mathrm{D} \in \mathbb{D}$ in this list is called the *rank* of D. The rank of the relevant document is denoted as $r \in \mathbb{N}$.

Having a single relevant document for each query, the top-K evaluation metric gives the proportion of queries for which $r \leq K$ for a given $K \in \mathbb{N}$. Furthermore, we report the mean reciprocal rank (MRR), which is the average of $1/r$ over all queries.

Table 6.3: Retrieval results for our dataset (consisting of 2045 themes).

	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
C_{IIR}	0.470	0.593	0.648	0.699	0.792	0.531
C_{MEL}	0.231	0.363	0.430	0.500	0.599	0.299
C_{SFM}	0.742	0.818	0.839	0.863	0.894	0.779
C_{BG1}	0.754	0.835	0.861	0.885	0.913	0.792
C_{DNN1}	0.417	0.534	0.576	0.633	0.708	0.474
C_{DNN2}	0.552	0.661	0.701	0.748	0.800	0.605
C_{CNN}	0.693	0.788	0.823	0.853	0.896	0.739
C^{*}_{MEL}	0.421	0.522	0.574	0.630	0.714	0.474
C^{*}_{BG1}	0.734	0.816	0.843	0.867	0.899	0.774
C^{*}_{CNN}	0.680	0.773	0.802	0.837	0.881	0.724

6.4.2 Retrieval Results

In this study, we want to focus on the aspect of monophonic–polyphonic matching. In [10], it was shown that factors such as tuning, transposition, and query length have a major impact on the retrieval results. To reduce the effect of these factors, we modify each MIDI query such that its duration and key matches the corresponding audio excerpt in the database. We could reproduce the results reported in the previous study [10] for their smaller dataset based on 177 queries (best reported top-1 rate 0.684) using the conventional chroma representation C_{IIR}. However, in our experiments with the smaller dataset, we could achieve a significant improvement compared to prior work, reaching a top-1 rate of 0.876 by using the enhanced feature representation C_{BG1}. This increase means that about 19 % more queries of the smaller dataset achieve a top-1 rank due to the improved feature representation.

We now systematically analyze the impact of the various time–frequency representations on the retrieval results by considering our larger dataset based on 2045 queries. The results are summarized in Table 6.3. The first row of the table shows the evaluation metrics for the baseline C_{IIR}. The top-1 rate (0.470) means that even for this simple approach, nearly half of the themes achieve a rank of 1. About 70 % of the queries yield a rank of at least 20 (top-20: 0.699). Harmonic summation performs worse, yielding a top-1 rate of 0.231 for C_{MEL}. We want to note that it is not fair to directly compare this representation with the others since harmonic summation amplifies many time–frequency components that do not belong to the predominant melody. The source-filter model C_{SFM} brings a major boost in performance with a top-1 rate of 0.742 and a top-20 rate of 0.863. C_{BG1}, which is a combination of harmonic summation and the source-filter model, further increases the top-1 rate to 0.754. About three-quarters of the 2045 themes yield a rank of 1, and about 89 % achieve a rank of at least 20 (top-20: 0.885), which is a major improvement compared to the baseline approach. The fully connected network C_{DNN1}, trained on jazz music, falls below the baseline for this task with a top-1 rate of 0.417. However, the version C_{DNN2}, which was re-trained on classical music, shows an increase of about 14 % in the top-1 rate (0.552) compared to the original version of the network. This increase reconfirms the obvious fact that such neural networks are highly data-dependent. Note that both networks have a simple architecture and do not exploit temporal

context. The more advanced neural network approach C_{CNN}, which involves convolutional layers covering more audio context, performs better than the simple networks. C_{CNN} achieves a top-1 rate of 0.693 and is close, but yet worse than the best model-based approaches C_{SFM} (0.742) and C_{BG1} (0.754).

In further experiments, we also performed melody extraction as an intermediate step, which turned out to typically worsen the overall retrieval result for our Western classical music scenario. Only C_{MEL} improved (an increase of top-1 rate from 0.231 to 0.421 with C^{*}_{MEL}), but this result is still below the baseline. In the case of the other representations, the results slightly worsen, i.e., we observe a drop of top-1 rate from 0.754 (C_{BG1}) to 0.734 (C^{*}_{BG1}) or from 0.693 (C_{CNN}) to 0.680 (C^{*}_{CNN}).

As another contribution of this chapter, we set up an extensive website presenting the ranks and sonifications for all 2045 queries, as well as visualizations of the corresponding feature representations.[31]

6.4.3 Discussion of Matching Quality

The rank-based evaluation metrics are rather coarse indicators of the matching quality. In the following, we want to examine it on a more fine-grained level. To this end, we introduce a novel evaluation metric called *separation indicator*. For a given query Q, we obtain an ordered list of documents $(\mathrm{D}_1, \mathrm{D}_2, \mathrm{D}_3, \dots)$. The position of the relevant document in this list is denoted as rank $r \in \mathbb{N}$. The matching quality is good when the matching cost of the relevant document $\delta^{\mathrm{Q}}_{\mathrm{D}_r}$ is significantly lower than the matching cost for the non-relevant document with the highest rank. The separation indicator $\rho \in \mathbb{R}_{\geq 0}$ quantifies the quality of the matching as follows:

$$\rho = \begin{cases} \delta^{\mathrm{Q}}_{\mathrm{D}_1} / \delta^{\mathrm{Q}}_{\mathrm{D}_2} & \text{if } r = 1, \\ \delta^{\mathrm{Q}}_{\mathrm{D}_r} / \delta^{\mathrm{Q}}_{\mathrm{D}_1} & \text{otherwise.} \end{cases} \tag{6.1}$$

Intuitively speaking, a low ρ below 1 implies a good matching quality since it indicates a distinct separation between the matching costs of the relevant document (with a rank of 1) and the first non-relevant document (with a rank of 2). The smaller ρ, the better the relevant document is separated from the non-relevant documents. A ρ close to 1 indicates that the top-1 decision is unstable for this query, and $\rho > 1$ implies that the query Q does not achieve rank 1. The distribution of ρ values across all queries is a good indicator of the stability of top-1 decisions.

Figure 6.2a and b show histograms of ρ values for the baseline C_{IIR} and the best performing representation C_{BG1}, using a logarithmic scaling of the horizontal axis. For C_{IIR}, the ρ values are centered around 1, indicating instability for the top-1 decisions. For C_{BG1}, the distribution is skewed to the left, which indicates that the increase in the top-1 rate of C_{BG1} is not due to small random-like changes of the matching costs, but to a substantial improvement in matching quality. Figure 6.2c shows boxplots of the distributions of ρ values for all considered feature representations. In this figure, the first and fourth rows correspond to the histograms shown in Figure 6.2a and b, respectively. For C_{MEL} (second row), the distribution is strongly centered just above 1.0, meaning that most queries do not achieve a rank of 1. The corresponding

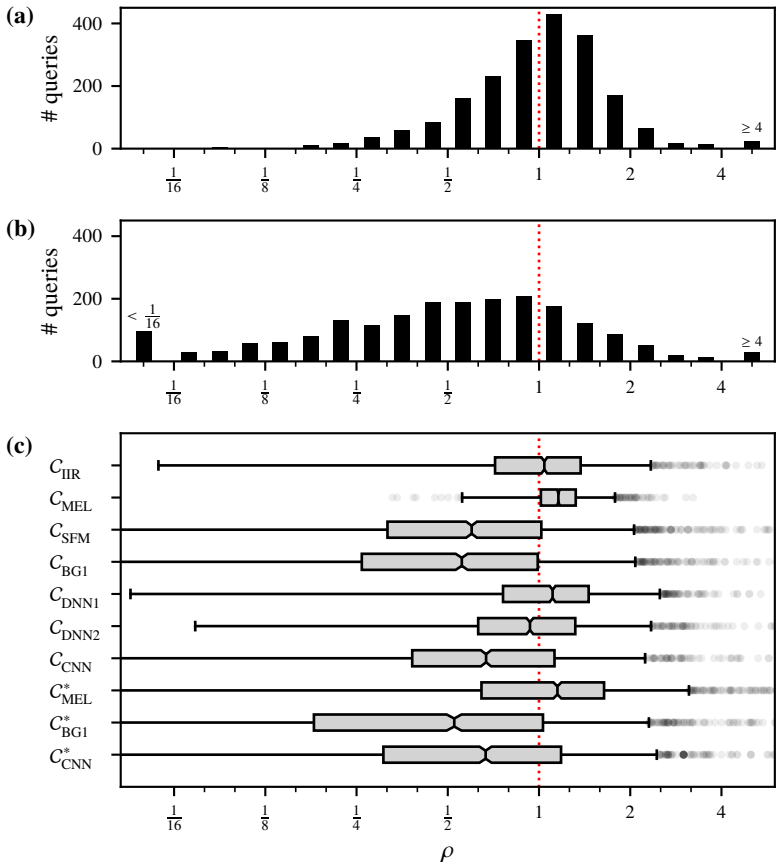


Figure 6.2: **(a)** Histogram of ρ values for C_{IIR} and **(b)** C_{BG1}. **(c)** Boxplots of ρ values for all representations. Queries to the left of the red line ($\rho = 1$) yield a top-1 match.

melody extraction C^*_{MEL} (eighth row) has a strong effect on the distribution: Though the median is located to the right of the red line (i.e., less than half of the queries achieve rank 1), it shows that the queries cover a wider range of matching qualities. The best performing representations C_{SFM}, C_{BG1}, and C_{CNN} (third, fourth, and seventh row) show similar tendencies: Most queries are located on the left side of the red line ($\rho < 1$) and cover a wide range of ρ values in that region, including queries with a high matching quality of $\rho < 1/2$. The comparison of C_{BG1} and C^*_{BG1} reveals that melody extraction improves the separation indicator for many queries. However, the top-K rates are worse for C^*_{BG1}. This may be explained as follows: For cases where the music has a low degree of polyphony, the intermediate melody extraction step lowers the separation indicator significantly. But, in more complex cases, the step does more harm than good.[33] As an effect, the overall retrieval result is better without melody extraction.

6.5 Conclusions

In this chapter, we considered a cross-modal retrieval scenario with the goal to find the relevant audio recording in a database, given a monophonic symbolic theme of Western classical music as a query.

[33] See `https://www.audiolabs-erlangen.de/resources/MIR/2019-ICASSP-BarlowMorgenstern/example` for two illustrative examples.

Extending previous work [10], we showed that salience representations, originally designed for melody extraction, are suited for the given task. Furthermore, unlike related work [166], we showed that in our retrieval scenario, it is beneficial to avoid an explicit melody extraction step and to perform the chroma reduction directly on the salience representations. In an extensive quantitative study, we compared various state-of-the-art salience representations and showed their benefits and limitations for the given task. Especially, the salience representation by Bosch and Gómez [26] turned out to be well-suited for monophonic–polyphonic matching. The results of our experiments have been made available on an accompanying website.[31] The following chapter will deal with the design of specialized representations for further improving the retrieval results.

7 CTC-Based Learning of Deep Chroma Features for Cross-Modal Music Retrieval

This chapter is based on [215] and [217]. Frank Zalkow is the first author of and the main contributor to both papers. In collaboration with his supervisor Meinard Müller, Frank Zalkow formalized the problem, developed the ideas, and wrote the papers. Furthermore, Frank Zalkow implemented the approaches and conducted the experiments.

In the previous chapter, we adapted several melody-enhanced pitch salience representations from the literature for a cross-modal music retrieval task where we aim to find relevant audio recordings of Western classical music, given a short monophonic musical theme in symbolic notation as a query. Our retrieval strategy was based on chroma features, which capture melodic and harmonic properties. In this chapter, we propose to learn a task-specific chroma variant for the theme-based retrieval task. Several recent studies demonstrated the effectiveness of DNNs that learn task-specific mid-level representations for different music retrieval applications. Usually, such supervised learning approaches require score–audio pairs where individual note events of the score are aligned to the corresponding time positions of the audio excerpt. However, in practice, it is tedious to generate such strongly aligned training pairs. In this chapter, we use weakly aligned training pairs, where only the time positions of the beginning and end of a theme occurrence are annotated in an audio recording, rather than requiring local alignment annotations. As one contribution, we show how to apply the connectionist temporal classification (CTC) loss in the training procedure, which only uses such weakly aligned training pairs. We evaluate the resulting features in our theme retrieval scenario and show that they improve the state of the art for this task. As a main result, we demonstrate that with our CTC-based training procedure using weakly annotated data, we can achieve results almost as good as with strongly annotated data. Furthermore, we assess our chroma features in depth by inspecting their temporal smoothness or granularity as an important property and by analyzing the impact of different degrees of musical complexity on the features.

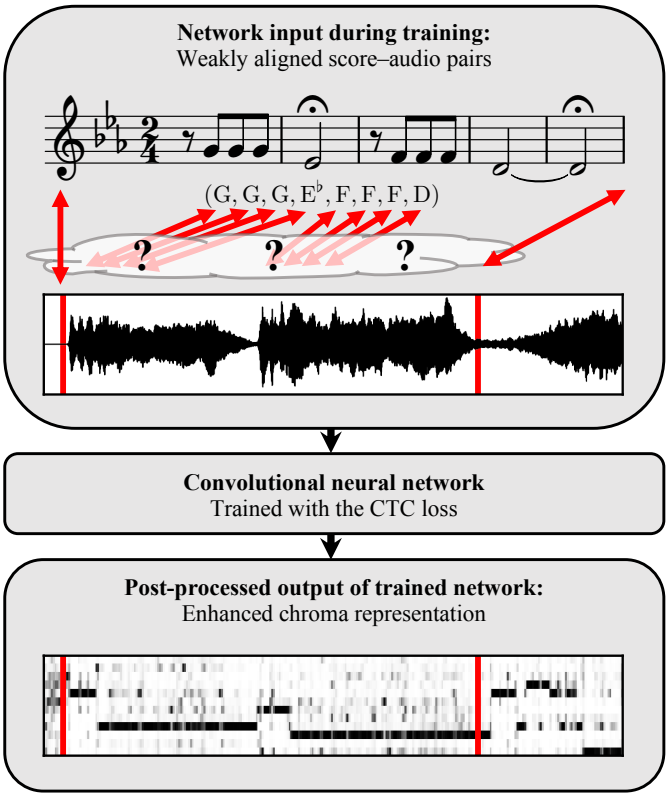


Figure 7.1: Illustration of a weakly aligned score–audio pair as the input to a CNN during training, and an enhanced chroma representation as the post-processed output of the network after training. Music example: First movement of Beethoven's Fifth Symphony, first theme.

7.1 Introduction

Audio recordings and symbolic scores are important music representations in many MIR tasks. A typical MIR application is cross-modal retrieval, where a symbolic score is given as a query, and the task is to identify relevant audio recordings [10, 81, 143, 188, 221]. In this chapter, we use monophonic musical themes in symbolic encodings as queries and aim to find all recordings, where these themes are played, in an audio database of Western classical music [10, 221]. A famous theme is, for example, the beginning of Beethoven's Fifth Symphony, which we show in Figure 7.1 in a score representation and as a waveform of a performance. Our retrieval scenario is challenging due to various reasons. One important aspect is that the queries are monophonic, but the themes usually appear with additional musical voices in the audio recordings.

Typical cross-modal retrieval strategies employ a common mid-level representation to compare the different modalities. In traditional music processing, chroma features are widely used as mid-level representation [14, 68, 128]. These features, which capture the energy in the twelve chromatic pitch class bands, are robust to a certain degree against changes in octave, instrumentation, and timbre. In general, computing chroma features with traditional signal-processing techniques involves many design choices. As our main contribution, we learn a task-specific chroma representation from training data.

Several studies have shown the benefits of deep-learning models to compute enhanced mid-level representations [23, 94, 95, 161]. These learned features have proven their effectiveness in many scenarios, such as audio–audio retrieval [48, 211, 216], chord recognition [94, 95, 206], or melodic pitch tracking [11, 15, 23]. Usually, training DNNs for these tasks requires aligned training pairs of audio recordings and corresponding annotations where, for each time position (or frame) of the audio recording, one has an annotation (or class label) to be learned by the model. We denote such pairs as "strongly aligned." For example, recordings with strongly aligned chord annotations have been used to train DNNs for computing chroma-like mid-level features for chord recognition [94, 95]. In general, strongly annotated datasets are crucial for learning meaningful music representations [22, 82]. However, creating strongly aligned training pairs is labor-intensive, and, for many music scenarios, data of this kind is hardly available. Rather than providing local alignments, it is much easier to annotate global correspondences, i.e., the beginning and end time positions of an annotated segment. We denote globally corresponding training pairs without local alignment as "weakly aligned." Figure 7.1 illustrates such a weakly aligned score–audio pair for our Beethoven example, where the beginning and end time positions of a theme occurrence are annotated in an audio recording without local alignment in between.

To utilize such weakly aligned training data, we use the *connectionist temporal classification* (CTC) loss to train a neural network. Graves et al. [71] originally introduced this loss for labeling unsegmented feature sequences with recurrent DNNs in the context of speech recognition. In the CTC training procedure, a kind of local alignment is computed as part of the loss function rather than having alignment annotations in the training data. We train a neural network with the CTC loss to compute enhanced audio chroma features (see Figure 7.1), which are as close as possible to the chroma representations of the symbolic themes. Our network architecture is inspired by a CNN that was originally used to compute melodic pitch salience representations [23].

As a main result, we demonstrate that the CTC-based features improve the state of the art for our cross-modal theme retrieval application. As another major contribution, we compare our CTC-based results (using weakly annotated data) with results obtained by a standard training strategy (using strongly annotated data). We show that the retrieval quality achieved with our CTC-based model is almost as high as with a model that we trained with strongly aligned data. We conclude that one can save a lot of tedious annotation work and access much more training data easily by using the CTC loss. To get a deeper understanding of these results, we present several quantitative and qualitative analyses of our CTC-based chroma representation. We show that different features have distinct properties in terms of temporal granularity (or smoothness) and that these properties have an impact on the retrieval results. We also investigate how the musical texture (such as monophonic, homophonic, and polyphonic) affects the retrieval results.

In modern MIR research, reproducibility is an important aspect [120]. To make our results transparent and accessible, we provide a website with various tools and interfaces.[34] First, we make all details of our

[34] `https://www.audiolabs-erlangen.de/resources/MIR/2020_IEEE-TASLP-ctc-chroma`

retrieval results available on an interactive web interface. Second, we provide pre-trained models and code to apply them. Third, we use the MTD (described in Chapter 5) as training data, which is publicly accessible.

The remainder of the chapter is organized as follows. In Section 7.2, we discuss our cross-modal retrieval application, including related work (Section 7.2.1), our dataset (Section 7.2.2), and our retrieval approach (Section 7.2.3). Then, in Section 7.3, we review deep salience and deep chroma models used in previous work (Section 7.3.1) and our study (Section 7.3.2). We then describe the CTC loss used to train our network in Section 7.4. Next, in Section 7.5, we present an evaluation of our CTC-based features in the context of our retrieval application. As a further main contribution, we analyze in Section 7.6 the effect of our CTC strategy by comparing it with standard approaches to train neural networks. Furthermore, in Section 7.7, we analyze the features' temporal granularity. Then, in Section 7.8, we come back to our retrieval application and analyze the impact of musical complexity on the retrieval results. Finally, we conclude with Section 7.9.

7.2 Cross-Modal Retrieval Application

Closely following Chapter 6, this section describes our cross-modal retrieval application, which is our motivating scenario for learning enhanced chroma features. Readers familiar with cross-modal retrieval may skip this section at first reading if they want to get straight to our main contributions. In the following, we review related work on theme-based retrieval, describe our dataset, and outline a typical retrieval pipeline. This basic retrieval procedure is used later in the experiments to evaluate our learned features.

7.2.1 Related Work

In cross-modal music retrieval scenarios, the aim is to find correspondences between different types of music representations [136], such as audio recordings, symbolically, or graphically encoded sheet music. For example, an audio-visual retrieval task is to find audio excerpts that match a given graphical sheet music representation (or vice versa) [53]. In our retrieval application, we aim to find relevant audio recordings for a given symbolically encoded musical theme as a query. Several studies already addressed theme-based music retrieval [10, 145, 215, 221]. A previous study [10] pointed out the challenges of the task, which are due to the differences in modality (symbolic vs. audio), tuning, transposition, tempo, and musical texture between the query and the recordings. The difference in musical texture is a major challenge because the themes are monophonic, but they usually appear in a polyphonic context in the recordings. Our experiments of Chapter 6 have shown that pitch salience representations are suitable mid-level features to compare the audio recordings with the symbolic themes. In this chapter, building upon these findings, we introduce an approach for learning a task-specific salience representation for our theme-based retrieval task.

Table 7.1: Dataset overview.
Duration format: hh:mm:ss.

Themes			Audio Recordings		
#	Mean Dur.	Total Dur.	#	Mean Dur.	Total Dur.
2067	00:00:09	04:57:48	1126	00:06:24	120:03:03

7.2.2 Dataset

For our experiments, we use the MTD described in Chapter 5. For each theme, the MTD provides a symbolic encoding and an occurrence in an audio recording. Furthermore, it comprises annotations about differences in transposition between the symbolic and audio versions. For our retrieval experiments, we also use the entire audio recordings, where the occurrences have been annotated. In total, the audio database consists of 1126 audio recordings with a duration of about 120 hours. A theme corresponds to precisely one recording, which, in turn, can contain the rendition of several themes. Table 7.1 shows some statistics for our dataset.

7.2.3 Basic Retrieval Procedure

We use the same retrieval procedure as described in Section 6.4.1 to evaluate our learned chroma features. For convenience, we summarize our retrieval pipeline and our evaluation measures. First, we have a set of symbolic encodings of musical themes, which serve as *queries*. Furthermore, we have a collection of audio recordings, which we denote as database *documents*. For each query, there is exactly one audio document that contains a globally corresponding rendition of the query theme (i.e., matching duration and transposition). For a fixed symbolic query, the aim is to retrieve the corresponding audio document. To compare the query with a document, we convert both into chroma sequences. For the symbolic query, we simply compute a binary chroma representation. For converting the audio recording, we employ an enhanced chroma representation (from our CTC or a baseline approach, as described later). Then, we use SDTW to compare the query with subsequences of the document [128]. Inspired by [10, 221], we use the cosine distance, the step size condition $\Sigma := \{(2, 1), (1, 2), (1, 1)\}$, as well as the weights $w_{\text{vertical}} = 2$ and $w_{\text{horizontal}} = w_{\text{diagonal}} = 1$. As a result of SDTW, one obtains a matching function, where local minima point to locations with a good match between the query and a document subsequence. We consider the minimal value of the matching function as the distance between query and document.

To solve the retrieval task, we compute distances between all documents and the query. We then order the documents according to ascending distance values. A document's position in this ordered list is called the *rank* $r \in \mathbb{N}$ of the document. Recall that our dataset contains exactly one relevant recording for each query. The top-K evaluation metric yields a value of one if this relevant document is among the top K matches (i.e., $r \leq K$), and zero otherwise. We then average this metric across all queries. Furthermore, we report the mean reciprocal rank (MRR), which is the average of $1/r$ across all queries.

7.3 Deep Salience and Deep Chroma Models

This section summarizes deep salience and deep chroma models as used in previous work. We then describe our adaptation of a deep salience model to compute an enhanced chroma representation for our cross-modal retrieval application.

7.3.1 Related Work

In MIR, many studies have demonstrated the effectiveness of using deep-learning models to compute task-specific feature representations [83]. One example is the use of deep salience models to compute enhanced time–frequency representations (measuring the saliency of frequencies over time) for tasks such as melody [11, 15, 23] or multi-pitch tracking [23]. Another example is the use of deep chroma models for computing enhanced chroma features (encoding the energy in the twelve chromatic pitch class bands) for chord recognition [94, 95, 117, 206]. This chapter is inspired by the deep salience approach by Bittner et al. [23], who introduced a feature representation named HCQT as input for a CNN (see Section 2.5 for more details). The HCQT is a three-dimensional tensor, where the three dimensions are time, frequency (logarithmic scaling), and harmonics. The third dimension ensures that harmonically related frequency bins are neighbors across the depth of the tensor. This way, the convolutional kernels of the network can easily exploit harmonic frequency relationships. Many studies use this deep salience representation as a baseline [15, 186] or build upon this model for diverse tasks such as polyphonic fundamental frequency estimation [45], dominant melody estimation [161], instrument recognition [85], tempo estimation [1], or chord recognition [207].

The study of Wu et al. [207] is related to ours in two respects. First, they also use the HCQT representation, and, second, they use weakly aligned training data. However, they aim for chord recognition instead of learning a mid-level representation for cross-modal retrieval. In contrast to our contribution, they take a three-step approach: First, they use a pre-trained deep chroma extractor to compute features. Second, they automatically align their chord labels to the chroma features using an HMM. Third, they use a frame-wise DNN classifier for chord recognition. In this chapter, we present a single-step approach to realize the alignment within the DNN training procedure.

7.3.2 Deep Salience Model Adaptation

In this section, we explain our adaption of the deep salience model by Bittner et al. [23], who approached the task of melody and multi-pitch tracking using a strongly aligned dataset of 10 hours. In our case, we aim to learn an enhanced chroma representation for cross-modal retrieval, employing our weakly aligned 5-hour dataset of 2067 themes. To avoid overfitting to our smaller dataset, we simplified the original model in several ways by reducing the number of parameters and memory requirements. Additionally, we

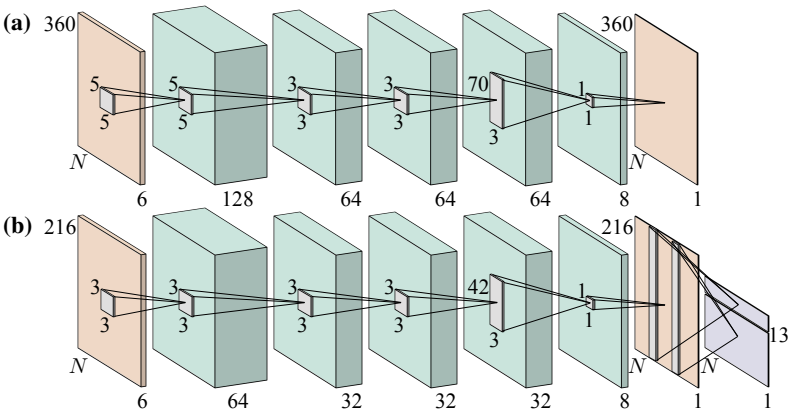


Figure 7.2: Network architectures. **(a)** Illustrations of the original architecture proposed by Bittner et al. [23] and **(b)** our adapted architecture used in this chapter. Illustrations inspired by [23]. **(c)** Details for our adapted architecture (72 970 parameters in total).

(c)

Layer	Output Shape	Activation	Parameters
Input	(N, 216, 6)		
Conv2D 64 × (3, 3, 6)	(N, 216, 64)	LReLU	3520
Conv2D 32 × (3, 3, 64)	(N, 216, 32)	LReLU	18 464
Conv2D 32 × (3, 3, 32)	(N, 216, 32)	LReLU	9248
Conv2D 32 × (3, 3, 32)	(N, 216, 32)	LReLU	9248
Conv2D 8 × (3, 42, 32)	(N, 216, 8)	LReLU	32 264
Conv2D 1 × (1, 1, 8)	(N, 216, 1)	Sigmoid	9
Pooling	(N, 13)	Softmax	217

adapted the network such that it can be trained with the CTC loss and used as a deep chroma extractor. Figures 7.2a and b illustrate the original network architecture and our adapted version, and the table in Figure 7.2c gives further details for our version. Compared to the model by Bittner et al. [23], we introduce the following modifications: First, we use a feature rate of 25 Hz instead of 86 Hz. Second, we use a frequency resolution of a third semitone instead of a fifth semitone. The high time and frequency resolutions of the original model may be beneficial in the application of melody estimation, but are not needed for our task (learning chroma features for retrieval). The decreased frequency resolution results in 216 instead of 360 frequency bins. As third modification, we reduced the number of filter kernels as well as the size of some of the filter kernels. The latter reduction accounts for the decreased frequency resolution. Fourth, we use leaky ReLU activations instead of ReLU activations to avoid zero gradients [113]. Fifth, we do not use batch normalization, which was used at the input to each layer in the original model. Instead, we ℓ^2-normalize all columns of the input to the network for being invariant to dynamics. Sixth, we add a pooling layer at the end, which we explain in the next paragraph.

After the last convolutional layer (with sigmoid activation), we obtain a representation that we could interpret as a kind of pitch salience of size $N \times 216$, where $N \in \mathbb{N}$ is the number of time steps. In our retrieval scenario, where we want to learn chroma-based mid-level features, we aim for an output size of $N \times 13$. Here, each of the N columns encodes a probability vector over the set of the twelve chroma labels and an additional blank symbol, which means that no chroma label is active (more details in Section 7.4.1).

Let us consider a single column of size 216 as input, which we want to transform to a probability vector of size 13. To compute the first twelve entries, we add up all pitch bins corresponding to the respective chroma bins. This fixed pooling has no learnable parameters. To compute the last entry for the blank symbol, we apply a standard dense layer (linear activation) to the input column. This layer has 217 learnable parameters (216 weights and a bias). Finally, we apply the softmax function to the resulting 13-dimensional vector. We transform all columns of the input using this pooling procedure.

In summary, our adapted model differs from the original model [23] in two important aspects: First, we reduced the number of parameters from 406 921 to 72 970. Second, the model's output is a sequence of probability vectors over 13 dimensions (encoding chroma vectors at a semitone-resolution) rather than 360 dimensions (encoding a pitch salience at a fifth semitone resolution).

7.4 CTC Loss

In this section, we introduce the CTC loss used to train our adapted model. Originally proposed for the task of speech recognition [71], CTC has been adopted to several MIR applications, including OMR [34, 35], monophonic audio-to-score transcription [160], lyrics alignment [74, 185, 197], and audio tagging [80]. An alternative to CTC for sequence learning without aligned training data is the usage of an attention mechanism, which was used for, e.g., monophonic singing voice transcription [137]. We now describe how to use the CTC loss for learning enhanced chroma features.

7.4.1 Loss Computation

In the following, we present the main idea of the CTC loss function introduced by Graves et al. [71]. We describe the CTC loss computation for a single pair consisting of an audio feature sequence and a label sequence. Let

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) \tag{7.1}$$

denote the feature sequence of length $N \in \mathbb{N}$, which consists of feature vectors $\boldsymbol{x}_n \in \mathbb{R}^D$ for $n \in [1 : N] := \{1, 2, \ldots, N\}$ and dimensionality $D \in \mathbb{N}$. The second sequence of the pair is a label sequence

$$\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M) \tag{7.2}$$

of length $M \in \mathbb{N}$, where typically $M \ll N$. This sequence consists of elements $\boldsymbol{y}_m \in \mathbb{A}$ for $m \in [1 : M]$. The alphabet $\mathbb{A}$ is the set of symbols that can occur in the label sequence. For example, in the case of lyrics alignment, the alphabet is the set of all possible characters [185]. In our case, it is the set of the twelve different chroma labels:

$$\mathbb{A} := \{\mathrm{C}, \mathrm{C}^{\#}, \mathrm{D}, \ldots, \mathrm{B}\}. \tag{7.3}$$

A DNN f_θ with parameters θ transforms the feature sequence $\boldsymbol{X}$ to a sequence of probability distributions

$$f_\theta(\boldsymbol{X}) = (p_1, p_2, \ldots, p_N) \tag{7.4}$$

having the same length N as the feature sequence. Each element of the sequence $p_n \colon \mathbb{A}' \to [0, 1]$ maps a symbol from the modified alphabet $\mathbb{A}'$ to a probability value. The modified alphabet

$$\mathbb{A}' \coloneqq \mathbb{A} \cup \{\epsilon\} \tag{7.5}$$

contains an additional blank symbol ϵ, which encodes that no symbol is active. We further explain the role of this symbol later in this section.

When we align the feature sequence $\boldsymbol{X}$ and the label sequence $\boldsymbol{Y}$, we assign a suitable symbol to each time frame. Intuitively, we can consider this alignment as an expansion of the label sequence $\boldsymbol{Y}$ to the length of the feature sequence. More formally, an alignment is represented by a sequence

$$\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N) \tag{7.6}$$

of elements $\boldsymbol{a}_n \in \mathbb{A}'$ and length N that satisfies the following condition. When removing all consecutive duplicates and then all blank symbols ϵ, the alignment sequence $\boldsymbol{A}$ is reduced to the label sequence $\boldsymbol{Y}$.

Given an alignment $\boldsymbol{A}$ and the sequence of probability distributions, we can compute the probability

$$P(\boldsymbol{A}|\boldsymbol{X}) = \prod_{n=1}^{N} p_n(\boldsymbol{a}_n) \tag{7.7}$$

of the alignment. When computing the CTC loss, we do not explicitly know the correct alignment, but only the label sequence. Because a label sequence can correspond to multiple alignments, all possible alignments between $\boldsymbol{X}$ and $\boldsymbol{Y}$ are considered. Let us denote the sum of probabilities for all these alignments as $P(\boldsymbol{Y}|\boldsymbol{X})$. The final CTC loss for a single training pair is

$$L_\theta(\boldsymbol{X}, \boldsymbol{Y}) = -\log P(\boldsymbol{Y}|\boldsymbol{X}). \tag{7.8}$$

This loss function is used in mini-batch gradient descent to update the parameters θ by averaging the loss value over all training pairs in a mini batch. By this procedure, the network's parameters improve to produce probability sequences that make the ground-truth label sequences of the training set more probable. Graves et al. [71] described how to compute $P(\boldsymbol{Y}|\boldsymbol{X})$ in a differentiable and efficient way using dynamic programming, similar to the forward algorithm for HMMs [76, 150]. We summarize this computation in Appendix B.

Finally, we want to clarify the role of the blank symbol ϵ. We already mentioned one role for ϵ, namely the indication of no active symbol, i.e., a rest in our music application. But as an additional role, the

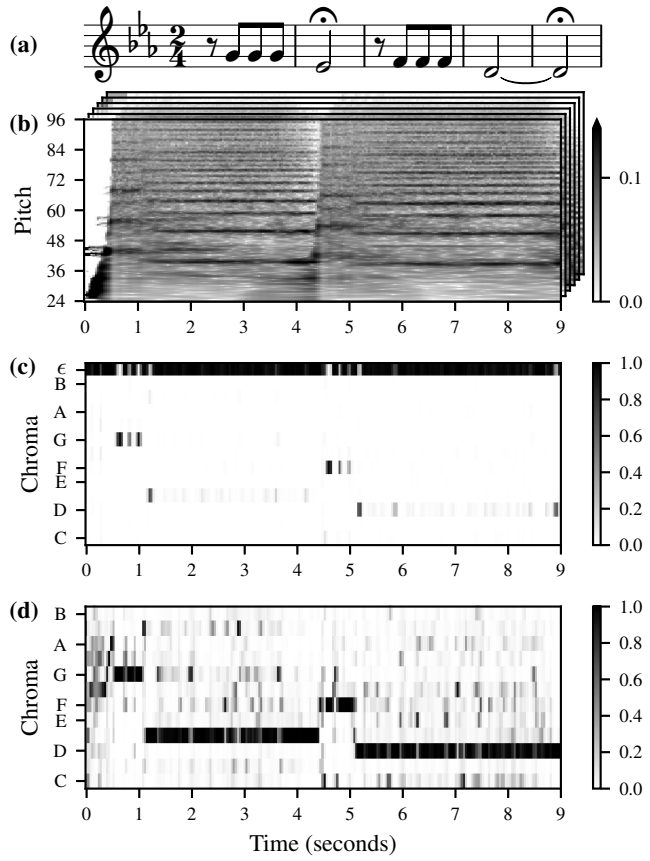


Figure 7.3: Representations for the first theme of Beethoven's Fifth Symphony. **(a)** Score of monophonic theme. **(b)** HCQT input representation $\boldsymbol{X}$ (front slice corresponding to the first harmonic). **(c)** Network output. **(d)** Features used for matching.

symbol also indicates repetitions. Let us illustrate this role using the first theme of Beethoven's famous Fifth Symphony as an example (Figure 7.3a). We only consider the beginning of the label sequence for brevity, i.e., $\boldsymbol{Y} = (\mathrm{G}, \mathrm{G}, \mathrm{G}, \mathrm{E}^\flat)$. An alignment of $\boldsymbol{A} = (\mathrm{G}, \mathrm{G}, \mathrm{G}, \mathrm{E}^\flat)$ would not correspond to this label sequence because we remove the consecutive duplicates when converting an alignment to a label sequence. Rather, this alignment corresponds to the label sequence $\boldsymbol{Y} = (\mathrm{G}, \mathrm{E}^\flat)$. To represent repeated symbols in the label sequence, we need to separate them by a blank. A valid alignment for the Beethoven excerpt is, e.g., $\boldsymbol{A} = (\mathrm{G}, \epsilon, \mathrm{G}, \epsilon, \mathrm{G}, \mathrm{E}^\flat)$.

7.4.2 Chroma Feature Computation

We train our model described in Section 7.3.2 with the CTC loss. The network's input is an HCQT tensor computed for an excerpt from an audio recording, where a musical theme is played. As an example, Figure 7.3a shows the score of our Beethoven theme. Figure 7.3b shows a slice of the HCQT features for a recording of this theme. The corresponding label sequence is the sequence of chroma labels of the theme with neither rhythmic information nor temporal alignment to the input (see also Figure 7.1). For this Beethoven example, the sequence is $\boldsymbol{Y} = (\mathrm{G}, \mathrm{G}, \mathrm{G}, \mathrm{E}^\flat, \mathrm{F}, \mathrm{F}, \mathrm{F}, \mathrm{D})$. The network's output is a sequence of probability distributions, visualized in Figure 7.3c for the Beethoven example. We see that the ϵ symbol has the largest probability most of the time, and the chroma labels only have large probabilities at the

Table 7.2: Retrieval results of the baseline methods **(a)** using a feature rate of 10 Hz as reported in Chapter 6, **(b)** using a feature rate of 25 Hz.

(a)

	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
C_{BG1}	0.754	0.835	0.861	0.885	0.913	0.792
C_{Bit}	0.693	0.788	0.823	0.853	0.896	0.739

(b)

	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
C_{BG1}	0.824	0.894	0.911	0.925	0.952	0.857
C_{Bit}	0.767	0.846	0.868	0.895	0.930	0.805

beginning of the corresponding note events. To use the network output as a feature representation, we remove the row corresponding to the ϵ symbol and interpret the resulting matrix as chroma features. Finally, we ℓ^2-normalize the 12-dimensional chroma vectors to compensate for the removed ϵ symbol. The ℓ^2 norm was chosen because the features are used later for SDTW, where the cosine distance is applied. Figure 7.3d shows the normalized chroma features, which correspond well to the label sequence.

7.5 Retrieval Experiments

We now evaluate the CTC-based chroma features in the context of our retrieval application. This task allows us to evaluate the features with quantitative evaluation measures. Using such measures, we also compare our learned representation with other features, computed by a traditional method and a deep-learning approach not adapted to the theme retrieval task.

7.5.1 Baseline Experiments

To compare our approach with prior work on cross-modal retrieval, we use the best-performing chroma representations from the study of Chapter 6. The first baseline chroma variant (C_{BG1}) is based on a traditional salience representation by Bosch and Gómez [26], which is computed by combining a source-filter model with harmonic summation, using threshold parameters that are particularly suited for orchestral music [25].[35] The second baseline chroma variant (C_{Bit}) is based on the original deep salience representation for melody estimation by Bittner et al. [23].[36] The network used to compute this representation was not adapted to our task and trained with a standard strongly aligned approach.

Table 7.2a cites the results from Chapter 6 (see Table 6.3 on p. 99), where a 10 Hz feature rate was used. According to this study, three quarters of the themes (75.4 %) yielded the relevant document on the first rank using C_{BG1}. For C_{Bit}, this is the case for 69.3 % of the themes. We checked whether the feature rate is appropriate for the given retrieval task and found that an increased time resolution is beneficial for the retrieval quality. A rate of 25 Hz turned out to be a good trade-off between retrieval accuracy

[35] The specific parameter setting is named "BG1" in [25].

[36] Original weights ("Melody 2"). C_{Bit} was denoted by C_{CNN} in Chapter 6. We rename it because we address several CNN-based representations in this chapter.

and efficiency. We repeated the experiments for C_{BG1} and C_{Bit} with the increased feature rate and show our results in Table 7.2b. Just by changing the temporal resolution, we see a substantial improvement in the results. For example, for C_{Bit}, the top-1 rate increases from 0.693 to 0.767, which means that the number of query themes with a correct top match increased by about 7 %. The reason for this may be the following: A fast tempo of *Presto* corresponds to up to 200 BPM. Having a quarter-note beat in such a tempo, a sixteenth note has a duration of 75 ms, which is shorter than the length of a frame given the feature rate of 10 Hz. In such cases, the increased feature rate is necessary to represent the musical content in a more meaningful way. For this reason, we use the feature rate of 25 Hz in all subsequent experiments.

Compared to the experiments of the previous chapter (which used a preliminary version of the MTD), the dataset was revised and slightly extended, accounting for up to 2 % improvements in the accuracy. Still, the main improvements are due to the increased time resolution.

For both feature rates, the representation C_{BG1} performs better than C_{Bit}. For example, the respective top-1 rates are 0.824 and 0.767 for the 25 Hz rate. The results for C_{Bit} may be lower because the training data of the underlying DNN consisted mainly of popular music (for overall 240 training tracks, only 22 are tagged as "classical" in version 1 of MedleyDB [22]). Another possible reason is that the saliency characteristics in the training data (coming from the "Melody 2" definition of MedleyDB) are different from the characteristics of musical themes.

7.5.2 Training Details

We split the 2067 score–audio pairs of our dataset (see Section 7.2.2) into five folds, where we use three folds for training, one for validation, and another one for testing. We ensure that all themes by a composer are part of precisely one fold. As a consequence, we do not use themes from the same composer for training and evaluation in order to avoid overfitting to the characteristics of particular composers. The first fold contains more themes (559) than the others because it contains all MTD themes by Ludwig van Beethoven, which is the most prominent composer of our dataset, as described in Chapter 5 (in particular, see Figure 5.3a on p. 83). The other folds have fewer themes (377) and are more diverse in terms of composers, having 12 to 14 different composers each.

During training, we apply circular shifts along the chroma axis as data augmentation to simulate transpositions (up to a minor third upwards and downwards). We perform mini-batch gradient descent with a mini-batch size of eight using the Adam optimizer [89] and a learning rate annealing procedure. In the first phase of this procedure, the initial learning rate is 0.001, and we train the model until the loss for the validation fold does not improve for five epochs. In the next phase, we halve the learning rate and continue the training with the model that has the lowest validation loss among the models of all previous epochs. We repeat ten such phases. When we finished training, we use the model with the lowest validation loss as a chroma feature extractor, and evaluate its effectiveness in the retrieval scenario. We only use the query

Table 7.3: Retrieval results for C_{CTC}.

Fold	Queries	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
1	559	0.875	0.941	0.957	0.964	0.970	0.903
2	377	0.828	0.891	0.905	0.926	0.947	0.859
3	377	0.859	0.918	0.934	0.952	0.966	0.884
4	377	0.899	0.947	0.958	0.968	0.981	0.922
5	377	0.873	0.931	0.950	0.958	0.981	0.900
⌀		0.867	0.927	0.942	0.955	0.969	0.894

Table 7.4: Retrieval results (⌀) for an oracle of the baseline by Bosch and Gómez [26] and our CTC approach.

	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
C_{BG1}	0.824	0.894	0.911	0.925	0.952	0.857
C_{CTC}	0.867	0.927	0.942	0.955	0.969	0.894
Oracle	0.907	0.947	0.960	0.970	0.983	0.925

themes from the respective test fold and all 1126 documents of our database for retrieval. The reported average evaluation measures (⌀) are weighted with the number of queries from the respective test fold.

7.5.3 CTC-Based Results

We now discuss the results we achieved with our CTC-based approach C_{CTC}. Table 7.3 shows the evaluation results for the five cross-validation iterations. The second column gives the number of query themes in the respective test fold. The retrieval results vary, ranging from a top-1 rate of 0.828 for the second test fold up to 0.899 for the fourth test fold. The last row of the table shows an average of the results, weighted by the number of queries used. Overall, we see a substantial improvement compared to the baseline approaches (Table 7.2b). For example, the average top-1 rate is 0.867 for C_{CTC}, compared to 0.767 for C_{Bit} and 0.824 for C_{BG1}. There are also improvements for larger ranks, such as in the top-50 rate (0.969 compared to 0.930 and 0.952, respectively). When repeating our training and evaluation procedures with different random initializations of the network weights, we only observed minor variations in the average evaluation measures (below 1 %). The results show that our approach is able to outperform the baselines, which are the state of the art for the given retrieval task [221].

7.5.4 Oracle Experiment

The traditional salience approach C_{BG1} also shows excellent performance for this task. To investigate the relationship between C_{BG1} and C_{CTC}, we evaluated both strategies with an oracle fusion procedure. For each query, we took the better rank: either achieved with the baseline or the CTC-based approach. Table 7.4 repeats the results for C_{BG1} and C_{CTC} for convenience and shows the oracle results in the third row. The oracle further improves the results for C_{CTC}. For example, the top-1 rate is 4 % larger (0.907 instead of 0.867). For top-K rates with larger K, there are still some small improvements. The oracle indicates that C_{BG1} and C_{CTC} behave differently on different queries by capturing different aspects. For

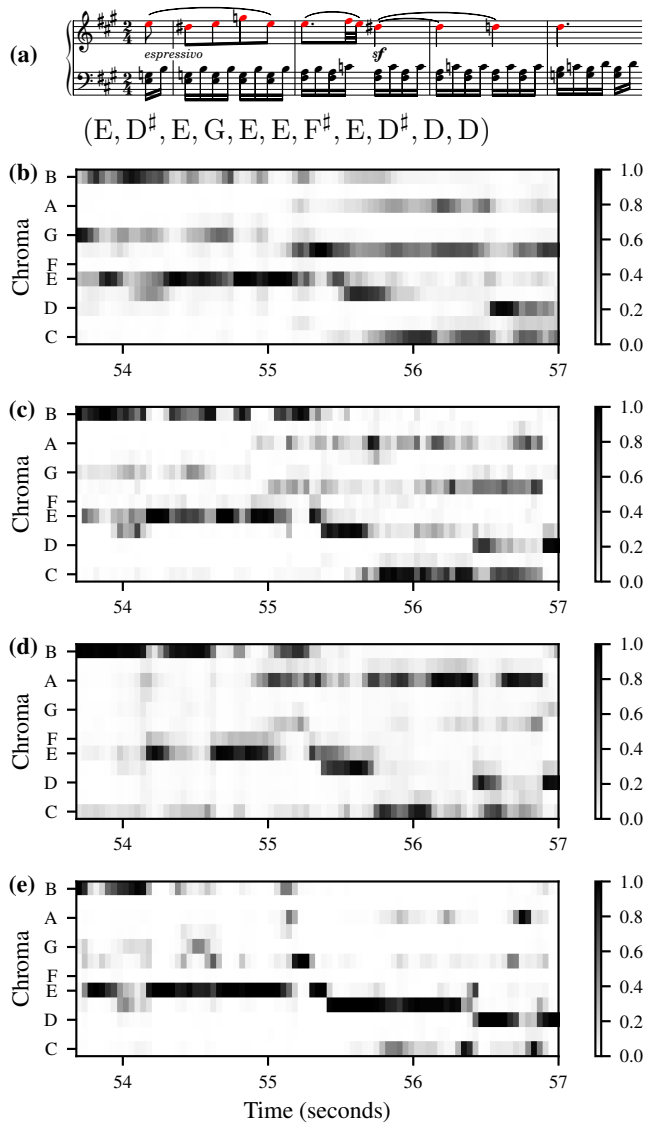


Figure 7.4: Second theme of Beethoven's Piano Sonata Op. 2, No. 2, first movement. **(a)** Full score with the theme's notes colored in red, along with the chroma sequence of the theme. **(b)** Standard chroma features using the full spectral content. **(c)** C_{BG1}. **(d)** C_{Bit}. **(e)** C_{CTC}.

some queries, C_{BG1} is a slightly better feature representation than the CTC-based approach. We conclude that there is still some room for improvement in the given retrieval scenario for future work, e.g., by fusing different feature representations.

7.5.5 Representative Example

To illustrate the properties of our CTC-based features, we close this section by comparing various mid-level representations for a representative example.

Figure 7.4a shows the full score and the chroma sequence for the second theme in the first movement of Beethoven's Piano Sonata Op. 2, No. 2. In this case, the theme is played by the right hand (upper staff), and the left hand (lower staff) plays an accompaniment. The sixteenth notes of the accompaniment

present a minor triad (E, G, B) in the pickup and first measure, and a diminished triad (F♯, A, C) in the second and third measure. Ideally, for our retrieval scenario, we aim for a chroma representation that only captures energy from the theme and not from the accompaniment. Figures 7.4b–e show chroma features for the full spectral content, the baseline salience approaches C_{BG1}, C_{Bit}, and our CTC strategy C_{CTC}, respectively. The accompaniment is strongly represented in the representation using the full spectral content (Figure 7.4b). For example, in the beginning, most energy is in the E, G, and B bands, which correspond to the accompaniment's E minor triad. In the representation C_{BG1} (Figure 7.4c), the main notes of the theme are well represented. However, some shorter notes of the theme (e.g., fourth note G or seventh note F♯) are not salient in this representation. C_{Bit} (Figure 7.4d) does not capture the theme well. This is especially the case in the second half, where the chroma bin A has the highest energy, which is part of the accompaniment. Among all representations, the theme is most evident in C_{CTC} (Figure 7.4e). In general, C_{CTC} attenuates the energy in the chroma bands corresponding to the accompaniment. The ability to represent the chroma energy of a musical theme is the main reason why our CTC-based features are a powerful tool for cross-modal melody-based retrieval.

7.6 Effect of CTC Loss

We showed that our CTC-based chroma representation outperforms the baseline approaches in our retrieval application. To further analyze the effect of the CTC strategy, we performed additional experiments, where we learned features without the CTC loss function. Instead, we trained our adapted DNN model using a standard loss function (categorical cross-entropy). This training procedure requires strongly annotated training data and implies classifying each spectral frame with respect to the twelve chroma labels.

7.6.1 Comparison with Linear Scaling

To treat our task as a classification problem, we have to assume a particular temporal alignment between the symbolic themes and the corresponding excerpts in the audio recordings. As a first alignment approach (C_{linear}), we assume a constant tempo throughout the theme occurrence. We used binary chroma representations as output labels by temporally scaling the symbolic themes to the same length as the corresponding audio excerpts in a linear way. Similar to the CTC strategy, this approach uses weakly aligned data, but we also used the rhythm information and note durations from the MIDI files (which we did not use for C_{CTC}). The blank symbol now only indicates rests in a theme. Since the assumption of a constant tempo is not realistic, we expect this to yield a sort of lower limit for the performance of our CTC-based approach.

We trained the model with the linearly scaled training data, used it as a chroma extractor, and then evaluated this strategy in the theme retrieval context. The first row of Table 7.5 repeats the average evaluation measures from Table 7.3 for convenience, and the second row presents the average results for

	Top-1	Top-5	Top-10	Top-20	Top-50	MRR
C_{CTC}	0.867	0.927	0.942	0.955	0.969	0.894
C_{linear}	0.829	0.897	0.914	0.929	0.953	0.863
C_{strong}	0.882	0.927	0.939	0.945	0.961	0.904

Table 7.5: Retrieval results (⌀) using cross-entropy.

the classification strategy with linear scaling (C_{linear}). For C_{linear}, the evaluation measures are lower than the CTC-based results, e.g., having a top-1 rate of 0.829 compared to 0.867. This difference is due to the non-linear temporal correspondence between the audio recordings and the symbolic themes.

7.6.2 Comparison with Strongly Aligned Data

As a second alignment approach (C_{strong}), we use manually aligned correspondences between the symbolic themes and the audio occurrences. In contrast to the previous strategies, this is a standard strongly aligned approach to train a neural network, similar to the training procedure used for the original deep salience model [23]. Creating the strong alignment annotations was highly labor-intensive because it implied annotating the onset time position for every note in each theme. We expect that training with strongly aligned data yields an upper limit for the performance of our CTC-based approach.

The third row of Table 7.5 shows the results for the classification approach using the manual alignments (C_{strong}). The strongly aligned approach slightly improves the results compared to the CTC strategy. For example, the top-1 rate is 0.882 compared to 0.867. For higher ranks, both strategies are on par with each other, e.g., yielding top-10 rates of 0.939 and 0.942, respectively. The fact that our CTC-based results are closer to the upper limit (C_{strong}) than the lower limit (C_{linear}) demonstrates that the CTC loss implicitly handles the alignment problem well in the training procedure. Without CTC, one has to take care of the alignment at the input level, using annotations, which are often not available or hard to generate.

7.6.3 Qualitative Comparison

Figure 7.5 shows the score and four audio feature variants for a monophonic theme by Bach. The first representation (Figure 7.5b) is based on a symbolic encoding, manually aligned to the audio occurrence. Figure 7.5c shows our CTC-based representation. Despite some noise, we see that C_{CTC} is similar to C_{MID}, which is not surprising for a simple monophonic theme. Figure 7.5d shows the representation C_{linear} from the weakly aligned classification approach, which is smoother compared to C_{CTC}. In fact, the representation C_{linear} is strongly over-smoothed. The features have this property because there is no accurate temporal correspondence between the input and output representations in the corresponding training strategy. As a consequence, the model temporally smears the active chroma bins. In the next section, we will further analyze the feature's temporal granularity and smoothness. The features from the

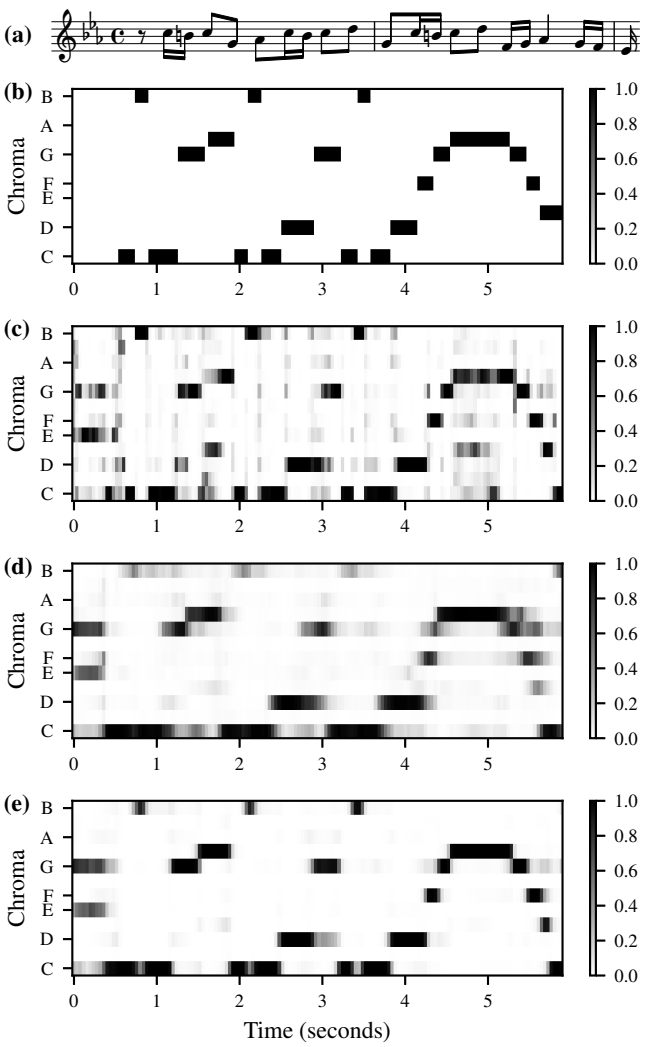


Figure 7.5: Feature representations for the theme of J. S. Bach's Fugue in C minor, BWV 847 (The Well-Tempered Clavier). **(a)** Score. **(b)** C_{MID}. **(c)** C_{CTC}. **(d)** C_{linear}. **(e)** C_{strong}.

strongly aligned classification approach C_{strong} (Figure 7.5e) are cleaner and sharper compared to the other two audio representations.

7.7 Temporal Granularity

In the previous section, we showed that differently learned features have distinct properties. In particular, we observed different degrees of temporal granularity and smoothness in the feature representations. To better understand how these differences impact our retrieval results, we now compare the chroma representations C_{CTC} and C_{BG1} in terms of temporal granularity.

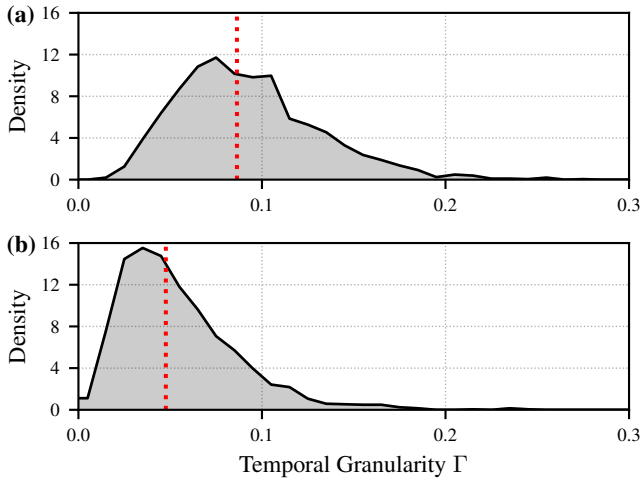


Figure 7.6: Distribution of granularity values for **(a)** C_{CTC} and **(b)** C_{BG1}. The red dotted lines indicate the median value of the respective distribution.

7.7.1 Granularity Measure

We now introduce a measure to quantify the temporal granularity of a feature representation on a scale from low granularity (i.e., smooth) to high granularity (i.e., fine-grained). To this end, we compare a sequence of ℓ^2-normalized chroma feature vectors

$$\boldsymbol{C} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_N) \tag{7.9}$$

of length $N \in \mathbb{N}$, having elements $\boldsymbol{c}_n \in \mathbb{R}^{12}$ for $n \in [1 : N]$, with a smoothed variant of $\boldsymbol{C}$. To compute this variant, we apply a temporal average filter of 6 frames (corresponding to 240 ms) and then again ℓ^2-normalize each vector. We apply the smoothing in a centric way, using suitable zero-padding conventions. As a result, the smoothed sequence

$$\boldsymbol{C}^{\mathrm{smooth}} = (\boldsymbol{c}_1^{\mathrm{smooth}}, \boldsymbol{c}_2^{\mathrm{smooth}}, \ldots, \boldsymbol{c}_N^{\mathrm{smooth}}) \tag{7.10}$$

has the same length N as the original sequence. Then, inspired by the distance measure used for SDTW, we compute as a measure of temporal granularity $\Gamma\colon\ \mathbb{R}^{12\times N} \to [0, 1]$ the average cosine distance

$$\Gamma(\boldsymbol{C}) = \frac{1}{N}\sum_{n=1}^{N}\left(1 - \frac{\langle \boldsymbol{c}_n, \boldsymbol{c}_n^{\mathrm{smooth}}\rangle}{\|\boldsymbol{c}_n\| \cdot \left\|\boldsymbol{c}_n^{\mathrm{smooth}}\right\|}\right). \tag{7.11}$$

High Γ values indicate fine-grained features since the features are dissimilar to their smoothed variants. Low Γ values indicate smooth features since they are similar to their smoothed variants. Figure 7.6 shows the distribution of Γ values for the 2067 themes of our dataset. For the CTC approach (Figure 7.6a), the median Γ is around 0.09, and for C_{BG1} (Figure 7.6b), it is approximately 0.05. This difference shows that the CTC approach is more fine-grained.

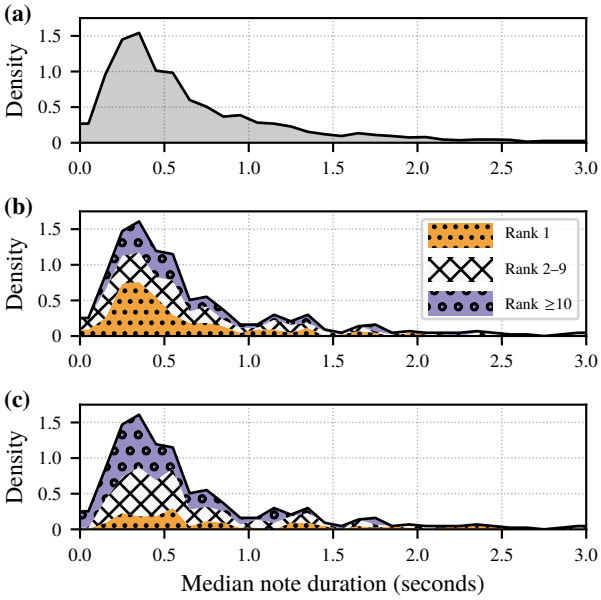


Figure 7.7: Distribution of median note durations for **(a)** all 2067 themes, **(b)** the set of difficult themes (color and hatches based on C_{CTC}), and **(b)** the set of difficult themes (color and hatches based on C_{BG1}).

7.7.2 Effect on Retrieval

How does the difference in temporal granularity of our features impact the retrieval results? A possible disadvantage of smooth features is that they hardly capture short note events. Since such events only span a short period, they may not be represented well in smooth features. Therefore, the CTC model's fine-grained features may perform better for themes with short note events. To examine this hypothesis, we relate the theme's note durations to the retrieval results. We compute the median note duration (using the manual alignments) for each theme and show the resulting distribution in Figure 7.7a. Most of the themes have a median note duration between 0.1 and 0.6 seconds.

Both feature representations C_{CTC} and C_{BG1} perform well in the retrieval application for many themes. For 1623 of the 2067 themes, the relevant document is ranked at the top (rank 1) in both approaches. We now consider only the remaining 444 themes yielding a non-relevant top match in at least one of the approaches. These themes constitute the more difficult part of our dataset. Only using this part, we again show the distributions of median note durations in Figures 7.7b and c. The distribution shape corresponding to the 444 themes is similar to the distribution shape corresponding to the total dataset (Figure 7.7a). The histogram's colors and hatches indicate the rank that a theme yielded in the two strategies C_{CTC} (Figure 7.7b) and C_{BG1} (Figure 7.7c). We see that most themes with a short median note duration of below 0.5 have better ranks in the CTC-based approach than for C_{BG1}. But, we have no substantial differences between the procedures for themes with a median note duration of above 0.5. Apparently, the fine-grained CTC-based features better represent themes with short note durations.

7.8 Musical Evaluation

After having analyzed the effect of the CTC loss and the features' properties, we now come back to our music retrieval application. A main challenge of the retrieval scenario is the fact that the queries are monophonic, and the audio recordings are polyphonic. In this section, we review categories of musical texture that help us to specify this monophony–polyphony discrepancy. Then, we analyze our retrieval results in terms of musical texture.

7.8.1 Musical Texture

We categorize the themes according to their musical texture, using the standard texture categories of monophony, homophony, and polyphony. We expect that more complex musical textures go along with decreased retrieval evaluation measures.

Closely following Chapter 5, we shortly review our used musical terminology. A monophonic texture consists of a single melodic line (possibly doubled by octaves). A homophonic texture is made up of a melody and an accompaniment. Themes with similar rhythm in all voices are also included into this category. A polyphonic texture comprises two or more independent melodic lines. Of course, different textures can appear in a single theme. For example, the beginning of a fugue often starts by presenting a musical theme in a monophonic way. Still, the texture turns polyphonic as soon as another voice joins in. For our analyses, we assign a single category to each theme, where we always use the most complex texture that occurs for that theme. Thus, the previous example (fugue theme starting monophonic and getting polyphonic later) is considered polyphonic. For musical examples and a further discussion on these categories, we refer to Chapter 5.

7.8.2 Analysis of Retrieval Results

Table 7.6 shows the evaluation results according to the categories monophony (M), homophony (H), and polyphony (P). For convenience, we also add results for the total dataset (T), which we already showed in Table 7.3. Let us consider the first row that reports on results for the first fold. This fold contains 40 monophonic, 418 homophonic, and 101 polyphonic themes. In total (T), 559 themes are in this fold. For 37 out of 40 monophonic themes, the relevant document was on the first rank (top-1: 0.93). 367 of the 418 homophonic queries had a relevant top match (top-1: 0.88). For the 101 polyphonic themes, this is the case for 85 themes (top-1: 0.84). For all 559 themes of this fold, the top-1 rate is 0.87. The evaluation rates negatively correlate with the complexity of the musical texture of the themes. This trend also holds for the average results ($\varnothing$), where the monophonic themes have a higher top-1 rate (0.93) than the homophonic themes (0.88) and the polyphonic themes (0.84). We can observe the same correlation for

		Queries				Top-1				Top-10				MRR			
	Fold	M	H	P	T	M	H	P	T	M	H	P	T	M	H	P	T
C_{CTC}	1	40	418	101	559	0.93	0.88	0.84	0.87	0.97	0.97	0.90	0.96	0.94	0.91	0.87	0.90
	2	11	308	58	377	0.82	0.84	0.74	0.83	0.91	0.91	0.86	0.90	0.87	0.87	0.79	0.86
	3	42	162	173	377	0.98	0.87	0.82	0.86	1.00	0.93	0.92	0.93	0.98	0.89	0.85	0.88
	4	16	236	125	377	0.88	0.91	0.88	0.90	0.94	0.95	0.97	0.96	0.89	0.92	0.92	0.92
	5	6	269	102	377	1.00	0.88	0.85	0.87	1.00	0.94	0.96	0.95	1.00	0.90	0.89	0.90
	⌀					0.93	0.88	0.84	0.87	0.97	0.94	0.93	0.94	0.94	0.90	0.87	0.89
C_{BG1}		115	1393	559	2067	0.95	0.83	0.79	0.82	0.97	0.91	0.89	0.91	0.96	0.86	0.83	0.86

Table 7.6: Evaluation on dataset parts with different musical characteristics: monophony (M), homophony (H), polyphony (P), total (T).

evaluation measures that take more than only the top rank into account. For example, the MRR is 0.94, 0.90, and 0.87 for the monophonic, homophonic, and polyphonic themes, respectively.

Similar trends can also be observed for C_{BG1} (last row). For example, the top-1 rates are 0.95, 0.83, and 0.79 for the monophonic, homophonic, and polyphonic themes, respectively. The CTC approach does not improves over C_{BG1} for monophonic themes (top-1: 0.93 and 0.95), but for the homophonic (top-1: 0.88 and 0.83) and polyphonic (top-1: 0.84 and 0.79) themes.

At first sight, it may be surprising that the difference between homophonic and polyphonic texture does not lead to greater differences in retrieval results. Though the difference in both categories is essential from a musical point of view, it may not be as important from a signal-processing perspective. In both texture categories, the theme appears with additional voices that make an audio occurrence dissimilar to its corresponding monophonic query. If these voices constitute an accompaniment (homophonic theme) or musically independent melodies (polyphonic theme) may only have a limited impact on our retrieval results. It is much more critical if the additional voices (independent or not) contribute a lot of energy to the theme occurrence in the audio recording.

7.8.3 Examples

To better understand the influence of the musical texture on our audio features, we discuss in the following several musical examples, which we order by increasing complexity. In Figure 7.5, we already showed a monophonic theme from a Bach fugue. There is no monophony–polyphony discrepancy between the query and corresponding audio occurrence for this theme. As a consequence, the query yielded a relevant top match in both C_{CTC} and C_{BG1} approaches. Figures 7.8 to 7.11 show four more musical themes in a sheet music representation of a piano reduction. The sheet music displays the theme's notes colored in red and all other voices in black. Along with the score, we show a symbolic representation (C_{MID}), which is manually aligned to the respective audio occurrence and, therefore, temporally corresponds to the audio features. Furthermore, we show the audio feature representations C_{CTC} and C_{BG1}.

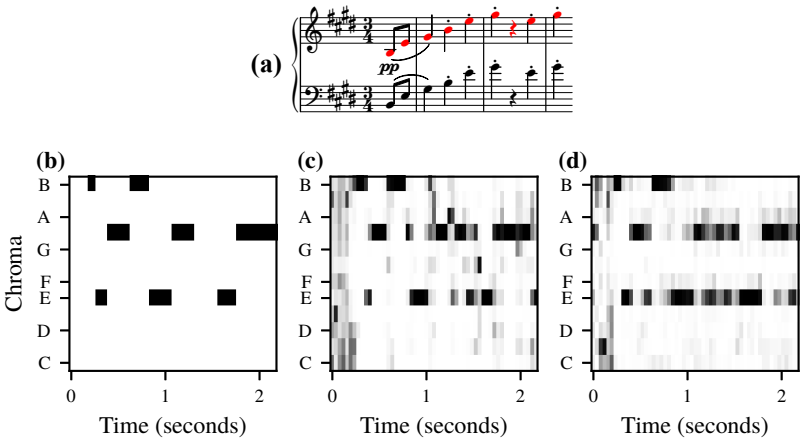


Figure 7.8: F. Schubert: Violin Sonata in A major, D. 574, second movement, beginning. **(a)** Sheet music of piano reduction. **(b)** C_{MID}. **(c)** C_{CTC}. **(d)** C_{BG1}.

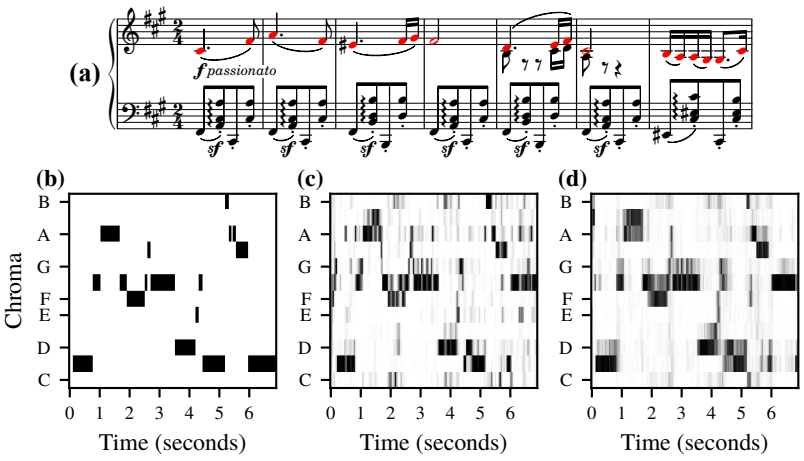


Figure 7.9: J. Brahms: Hungarian Dance No. 5 in F♯ minor, WoO 1, first theme. (We cyclically shifted the feature representations to match transposition of the sheet music.) **(a)** Sheet music of piano reduction. **(b)** C_{MID}. **(c)** C_{CTC}. **(d)** C_{BG1}.

The first example (Figure 7.8) is the beginning of a violin sonata by Schubert, where the theme is doubled by octaves. We still consider this monophonic. The feature representations capture the theme's pitch classes well, but we can still observe two problems. First, the rest (just before the second 1.5) is not well captured in both feature representations. In C_{CTC}, some unrelated pitch classes seem to be active, and in C_{BG1}, two previously active pitch classes (E and G♯) seem to be extended. The second problem is that the theme occurrence is short (2 seconds) and only contains three distinct pitch classes since the theme consists of a broken E major triad. As a consequence, the query is not very discriminative. Still, it achieved a rank of one and two in the approaches C_{CTC} and C_{BG1}, respectively.

The second example (Figure 7.9), which is homophonic, is the beginning of the famous Hungarian dance in F♯ minor by Brahms. Though the theme is accompanied, the theme's pitch classes are dominant in the feature representations. There seems to be a slight tuning issue in this example because many chroma bands share energy with their upper neighbor. The feature representations are still discriminative, and for this query, the relevant document was ranked at the top in both approaches.

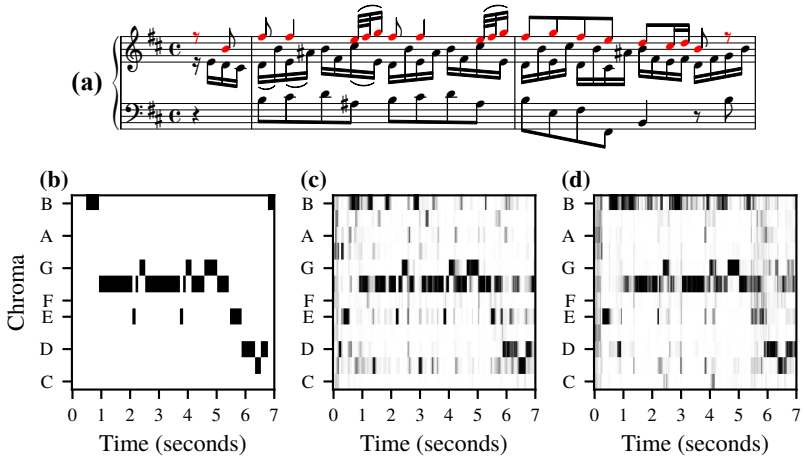


Figure 7.10: J. S. Bach: Flute Sonata in B minor, BWV 1030, beginning. (We cyclically shifted the feature representations to match transposition of the sheet music.) **(a)** Sheet music of piano reduction. **(b)** C_{MID}. **(c)** C_{CTC}. **(d)** C_{BG1}.

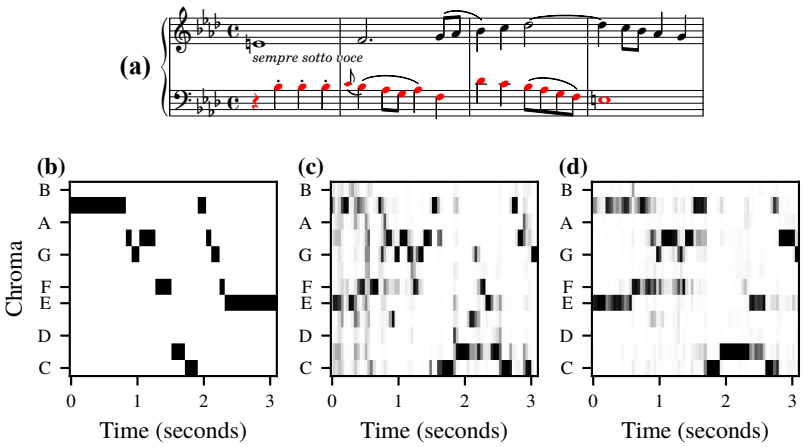


Figure 7.11: J. Haydn: Quartet in F minor, Hob. III:35, Finale, second fugue subject. **(a)** Sheet music of piano reduction. **(b)** C_{MID}. **(c)** C_{CTC}. **(d)** C_{BG1}.

The third example (Figure 7.10) is polyphonic and comes from a flute sonata by Bach. Though some energy from the other voices is present in the features (especially in the B band), the theme is well preserved in the feature representations. The theme yielded a relevant top match in both approaches.

The fourth example (Figure 7.11) is a subject from a string quartet fugue by Haydn. This is a complex case because two fugue subjects are overlapping. In this example, we consider the second subject played by the cello while the second violin presents the first subject (the second violin starts two measures before the beginning of this example). The feature representations contain both subjects to a certain degree, but the first subject is more strongly represented. The theme achieved the ranks of 42 and 6 in the approaches C_{CTC} and C_{BG1}, respectively.

Note that the retrieval results are generally good. To provide instructive examples, we over-emphasized problematic queries in the figure. On our accompanying website[34], we provide visualizations of all themes in all feature representations, which give a more complete picture. In summary, the examples show that our feature representations are well-suited for our application in many cases. However, we face problems in some cases due to many different reasons, such as few pitch classes, tuning issues, additional voices with high energy, or musical ambiguities.

7.9 Future Work and Conclusions

7.9.1 Future Work

Our study offers several aspects for further development. For example, in our retrieval experiments, we excluded the challenges due to differences in transposition between the queries and corresponding audio documents, which could be taken into account by circularly shifting the chroma features [10] or using a learning procedure for computing transposition-invariant features [6]. Furthermore, our oracle experiment (Section 7.5.4) suggests a possible next step of combining our strategy with traditional salience approaches [26].

Another possible extension of our work is to modify the CTC learning algorithm. In our system, the blank symbol ϵ is dominant in the network's output. To reduce its dominance, we remove the ϵ-probabilities and then normalize the features. As an alternative solution, one may modify the CTC algorithm such that it does not use the ϵ symbol. The symbol's first role (indicating a rest) could be taken over by an explicit rest symbol in the label sequence. The second role (indicating repetitions) is not needed in our scenario because we are not interested in transcription (i.e., finding the most probable label sequence given the output probabilities) but learning powerful features for retrieval.

7.9.2 Conclusions

In this chapter, we showed how to apply the CTC loss to train deep chroma models with weakly aligned training data. In our theme retrieval scenario, we improved the state of the art by using features learned by such a model.

As a main contribution, we compared the CTC strategy with weakly and strongly aligned classification approaches. We found that the CTC strategy is superior to standard DNN training procedures for weakly aligned training data. The CTC results are only slightly worse than the results obtained by training approaches using strongly aligned data, which can be considered the ideal case in our scenario. This finding is of major importance because it is highly labor-intensive to create strong alignments. A CTC-based strategy allows for saving a lot of annotation work and only leads to a slight drop in retrieval quality. This potential of CTC is also relevant for other MIR tasks, where annotations are hardly available, such as melody estimation [167] or chord recognition [141].

As a further contribution, our analyses revealed that temporal granularity is an important property of the features and that smooth features do not capture short note events well. Furthermore, we also showed that the retrieval results correlate with the themes' musical complexity, though even for complex polyphonic themes, we generally achieve good retrieval results. In various examples, we showed that our task-specific chroma features implicitly reduce the degree of polyphony of the audio content, which makes them well-suited for our retrieval task.

We make our contributions reproducible and accessible in three different ways.[34] First, we provide an interactive web interface that shows detailed retrieval results for each theme and includes visualizations of the various feature representations. A tabular view allows ordering the themes according to the corresponding retrieval ranks, making it easy to find well-behaved and problematic queries. Second, we make pre-trained models and code to apply them available, which allows computing the CTC-based chroma features for arbitrary audio files. Third, our training data is publicly accessible, including an overview website with score visualizations and sonifications (see Chapter 5).

We think that the findings of this chapter are relevant beyond our retrieval scenario. The trend towards using ever-increasing annotated datasets is considered critical among researchers. Data efficiency is still an important topic in the age of deep learning [79]. The MIR community is also critically aware of the fact that large annotated datasets are not equally accessible by industry and academia [44]. A step towards solving this problem could be to use weakly annotated datasets, which are much easier to obtain. Our work shows that procedures for weakly aligned annotations can achieve results nearly as good as approaches using strongly aligned annotations, encouraging further adaptations and developments of such procedures.

8 Summary and Future Work

In this thesis, we addressed two different cross-version retrieval scenarios of Western classical music, where the respective scenarios led to different retrieval strategies and different approaches for learning task-specific audio representations. In our first scenario (Part I), we used short audio snippets as queries to find all recordings in a database that are based on the same musical work as the respective queries. Building upon a shingle-based approach, we focused on the efficiency of the retrieval procedure. We showed how to learn low-dimensional embeddings of audio shingles by using PCA- or DNN-based techniques to enable retrieval strategies based on standard index structures for nearest neighbor search (Chapter 3). In addition, we applied a recent graph-based indexing approach to substantially increase the retrieval speed, even for shingle embeddings with higher dimensionalities (Chapter 4). In our second scenario (Part II), we used monophonic symbolic musical themes as queries to find recordings in a database, where the respective themes are being played. For this challenging task, where query (monophonic symbolic theme) and database recordings (audio recordings of polyphonic music) are fundamentally different from each other, we focused on improving the retrieval results. We first introduced the MTD, a novel dataset for musical themes that is essential for evaluation and training procedures in the theme-based retrieval scenario (Chapter 5). Then, we proposed a retrieval strategy based on melody-enhanced pitch salience representations that improved the results for the theme-based retrieval task compared to previous approaches (Chapter 6). Finally, we further improved the results by learning a task-specific audio representation for musical themes that was computed by a DNN trained with the CTC loss, using weakly aligned score–audio pairs from the MTD (Chapter 7).

In the scenarios considered, we have gained various insights on how to increase the efficiency and quality of the retrieval. As for future work, the insights obtained in one scenario could be applied to the other scenario. A central result in the shingle-based scenario was the increase in search efficiency through graph-based index structures. The application of index-based nearest neighbor search was possible by using embedding approaches to compute fixed-size vectors representing local sections of an audio recording. One possible research direction is to improve the efficiency of the theme-based retrieval application in a similar way. In our approach for this scenario, the CTC-based DNN processed and enhanced variable-length input representations without modifying the temporal dimension. This process resulted in queries and database items of variable lengths. For dealing with this variability, we used a computationally expensive retrieval strategy based on sequence alignment to compensate for temporal differences between the query and the database recordings. One may extend the CTC-based training for

computing fixed-size vectors representing entire themes. For example, one option would be to employ an encoder-decoder architecture [187], where an encoder embeds the input sequence (audio occurrence of a theme) into a fixed-size vector. A decoder then transforms this embedding into an output sequence (which may be aligned to a symbolic theme during training as in the CTC loss). The intermediate embedding, which should represent the entire musical theme, could be used for retrieval, which may open up the possibility of applying fast nearest neighbor search strategies for theme-based retrieval.

Another direction for future work concerns the representations used as input for the DNNs. In the shingle-based scenario, we used specialized chroma features (CENS), having a low feature rate and a strong temporal smoothing. Due to these characteristics, there was no need to deal with temporal differences between corresponding training items (i.e., anchor and positive examples) during the training procedure. On the contrary, we used input representations having a higher temporal resolution in the theme-based retrieval scenario. To compensate for temporal differences between the corresponding training items (i.e., symbolic theme and audio occurrence), we performed an implicit alignment during the training procedure (using the CTC loss). Similarly, one may also perform an alignment during training in the first scenario. Using such an approach, the DNN may perform a data-adaptive smoothing that could alleviate the need for specialized pre-processing of the network input.

In conclusion, this thesis presented various strategies to increase the efficiency and improve the results in two cross-version music retrieval scenarios, employing methods from information retrieval, signal processing, and machine learning. We want to emphasize that the findings of this thesis are relevant beyond the specific retrieval scenarios considered. In many multimedia domains, increasing amounts of data pose various challenges, especially when dealing with time-dependent data streams. In particular, the questions of how to improve the search efficiency and how to employ weakly annotated training data are important in many multimedia applications. In this thesis, we provided possible solutions to these questions using two music retrieval tasks as example scenarios.

Appendix

A Frequency Responses Used in the STFT-Based Log-Frequency Spectrogram Computation

We introduced various time–frequency transforms with a logarithmic frequency grid in Chapter 2. In particular, we described an approach to rescale the frequency axis of the STFT (Section 2.3), a transform based on IIR filters (Section 2.4), and the CQT, which is based on time–frequency atoms (Section 2.5). For the IIR-based spectrogram and the CQT, we visualized and discussed the frequency responses of the filters used in the respective transforms (Figure 2.6 on p. 17 and Figure 2.8 on p. 19). In this appendix section, we provide similar visualizations for the STFT-based log-frequency spectrogram.

In Chapter 2, we used suitable interpolation techniques to rescale the STFT's linear frequency axis. In the following, we use an alternative approach for computing the log-frequency spectrogram from the STFT. In this approach, we add up the amplitude values of all frequency bins that correspond to the same pitch in each frame. There may be pitch bins that do not have any corresponding frequency bin in the STFT, which leads to empty bins in the resulting log-frequency spectrogram. To fill these empty bins, one may increase the STFT's frequency resolution. For a detailed account of the pooling procedure, we refer to the textbook by Müller [128].

We can consider the windowed complex exponentials used in the STFT as FIR filters (see Eq. 2.2 on p. 12). Using an STFT with a Hann window of about 23.2 ms length ($N = 512$, $F_s = 22\,050$), we visualize in Figure A.1a the frequency response of the windowed complex exponential for $k = 20$, having a center frequency of $k \cdot F_s/N \approx 861.3$ Hz. This frequency is close to the pitch $p = 81$, having a center frequency of $f(81) = 880$ Hz. For computing the pitch bin $p = 81$ in our pooling procedure, we aggregate the amplitude values of all STFT bins corresponding to a center frequency between $f(80.5) \approx 854.9$ Hz and $f(81.5) \approx 905.8$ Hz. In our setting, these are the bins for $k = 20$ and $k = 21$ (having center frequencies of 861.3 Hz and 904.3 Hz, respectively). The frequency responses of these bins are visualized in Figure A.1b. We can derive an aggregated frequency response for $p = 81$ by adding up the responses for $k = 20$ and $k = 21$. This sum is visualized in Figure A.1c.

Figure A.2a shows the aggregated frequency response for the pitch bin $p = 69$ used in the pooling procedure for computing an STFT-based log-frequency spectrogram, where we used a Hann window of about 23.2 ms length ($N = 512$, $F_s = 22\,050$). Due to its wide bandwidth, the response covers more than a

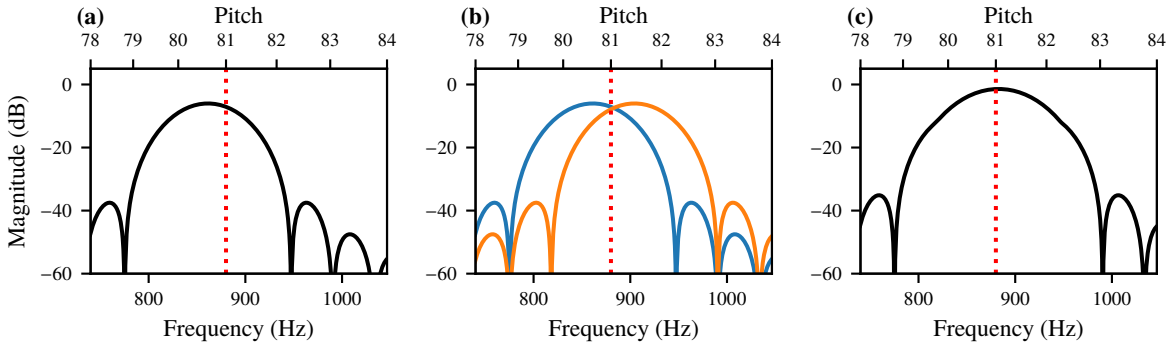


Figure A.1: Frequency responses for an STFT using $N = 512$ and $F_s = 22\,050$. **(a)** Frequency response of windowed complex exponential for $k = 20$. **(b)** Frequency response of windowed complex exponentials for $k = 20$ and $k = 21$. **(c)** Aggregated frequency response.

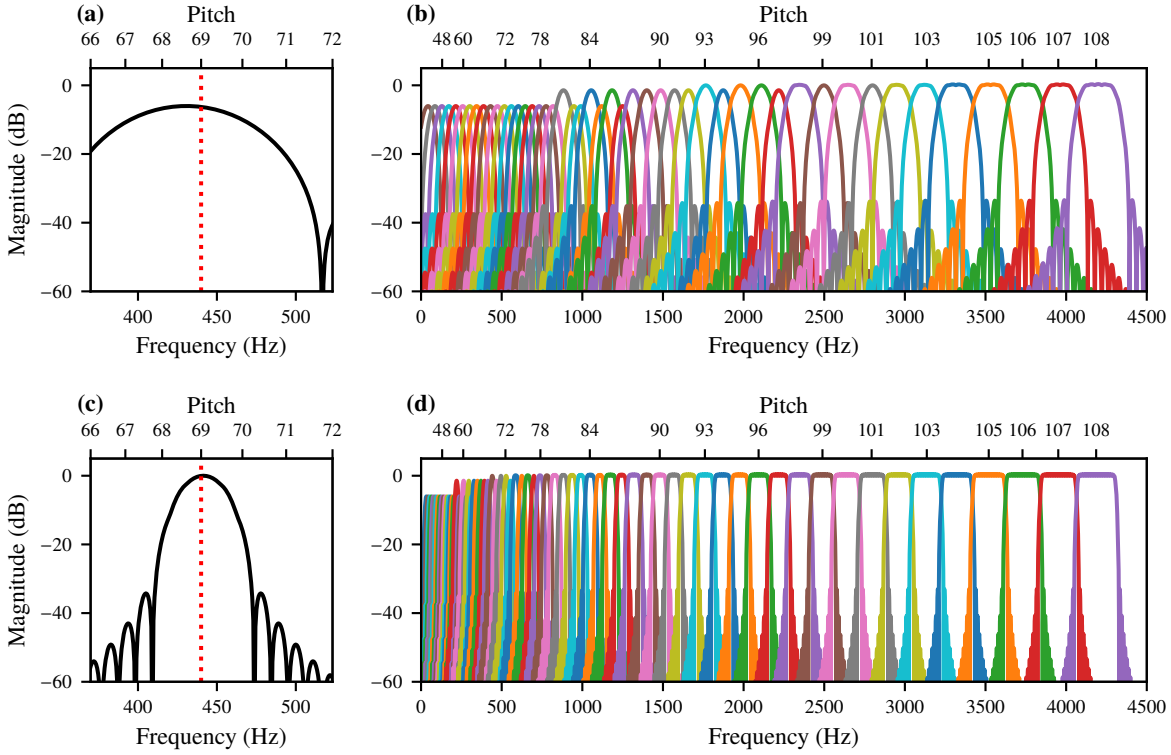


Figure A.2: Aggregated frequency responses used in the pooling approach to compute an STFT-based log-frequency spectrogram. **(a)** Frequency response for $p = 69$ ($N = 512$). **(b)** Frequency responses for $p \in [24 : 108]$ ($N = 512$). **(c)** Frequency response for $p = 69$ ($N = 2048$). **(d)** Frequency responses for $p \in [24 : 108]$ ($N = 2048$).

single semitone. When we increase the window length to 92.9 ms ($N = 2048$), the bandwidth becomes smaller (Figure A.2c). Here, the frequency response is similar to the corresponding response used in the CQT (displayed in Figure 2.8a on p. 19 in Chapter 2). Figure A.2b visualizes the aggregated frequency responses for the pitch bins $p \in [24 : 108]$ using $N = 512$. We see that lower pitches are not represented well, which is due to the linear frequency resolution of the STFT. The responses in the lower-pitch regions do not overlap visually as strong as we might expect because of the empty pitch bins discussed above. The

frequency responses with a passband peaking at around -6 dB only have a single associated frequency bin in the STFT. When more frequency bins are associated with a pitch, the passband of the respective frequency response has a higher peak. Figure A.2d visualizes the frequency responses using $N = 2048$. As expected, a higher window size, leading to a higher frequency resolution of the STFT, improves the characteristics of the bandpass filters. For example, the transition bands become shorter and lower pitches are resolved better.

B Connectionist Temporal Classification

B.1 Introduction

In Chapter 7, we described how we use the CTC loss [71] to train a CNN for computing chroma features. We outlined the general idea of the CTC loss computation in Section 7.4 without going into detail regarding the efficient dynamic-programming-based algorithm, which is used in practice to compute the loss. In this section, we describe this algorithm, following Graves et al. [71]. Before that, we also elaborate on the reduction of an alignment to a label sequence and the computation of the probability of a label sequence. Beyond the original article [71], we also recommend the review on the CTC loss in the thesis by Hannun [76]. In summary, we carefully adapted the mathematical notation style to highlight the relation to similar algorithms, such as dynamic time warping as described in the textbook by Müller [128].

B.2 CTC Loss Computation

We expect the reader to be familiar with the application and notation described in Chapter 7. For convenience, we shortly repeat our notation. Furthermore, we expand it to lay the foundations for the description of the dynamic-programming algorithm. We describe the CTC loss computation for a single training pair consisting of an audio feature sequence

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) \tag{B.1}$$

of length $N \in \mathbb{N}$ (consisting of feature vectors $\boldsymbol{x}_n \in \mathbb{R}^D$ for $n \in [1 : N]$ of dimensionality $D \in \mathbb{N}$) and a label sequence

$$\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M) \tag{B.2}$$

of length $M \in \mathbb{N}$ (consisting of elements $\boldsymbol{y}_m \in \mathbb{A}$ for $m \in [1 : M]$). The alphabet $\mathbb{A}$ is the set of symbols that can occur in the label sequence (such as the set of twelve different chroma labels). Typically $M \ll N$. The feature sequence $\boldsymbol{X}$ is transformed by a DNN f_θ with parameters θ to a sequence of probability distributions

$$f_\theta(\boldsymbol{X}) = (p_1, p_2, \ldots, p_N) \tag{B.3}$$

having the same length N as the feature sequence. Each element of the sequence $p_n \colon \mathbb{A}' \to \mathbb{R}$ maps a symbol from the modified alphabet $\mathbb{A}'$ to a probability value. The modified alphabet

$$\mathbb{A}' \coloneqq \mathbb{A} \cup \{\epsilon\}. \tag{B.4}$$

contains an additional blank symbol ϵ. Since the elements p_n are probability elements, their sum over the modified alphabet is one:

$$\sum_{\alpha \in \mathbb{A}'} p_n(\alpha) = 1, \tag{B.5}$$

for $n \in [1 : N]$. When we assign a suitable symbol to each time frame, we align the label sequence to the feature sequence. We can also think of such an alignment as an "unfolded" label sequence $\boldsymbol{Y}$. We denote an alignment by

$$\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N), \tag{B.6}$$

having the same length N as the feature sequence and consisting of symbols $\boldsymbol{a}_n \in \mathbb{A}'$. More formally, a valid alignment $\boldsymbol{A}$ can be converted to a label sequence $\boldsymbol{Y}$ using two simple rules. The first rule is that consecutive duplicate symbols are merged into a single symbol. We denote a function that realizes this rule by κ'. As an example, let us consider a sequence $\boldsymbol{A} = (\alpha, \alpha, \beta, \beta, \beta)$. When converting this sequence to a label sequence, it is reduced to $\kappa'(\boldsymbol{A}) = \boldsymbol{Y} = (\alpha, \beta)$. Having only this rule, we cannot express a repeated symbol in the label sequence $\boldsymbol{Y}$ by an alignment $\boldsymbol{A}$. To indicate repeated symbols in $\boldsymbol{Y}$, we need to separate the corresponding symbols in $\boldsymbol{A}$ by a blank symbol ϵ. A repetition is then realized by the second rule to convert $\boldsymbol{A}$ to $\boldsymbol{Y}$, which consists of removing all blank symbols from the sequence. We denote a function that implements the second rule by κ''. Applying both rules yields $\kappa = \kappa'' \circ \kappa'$. As an example, let us consider an alignment sequence $\boldsymbol{A} = (\alpha, \epsilon, \alpha, \alpha, \beta)$. After applying the first rule, the sequence is reduced to $\kappa'(\boldsymbol{A}) = (\alpha, \epsilon, \alpha, \beta)$. Then, after applying the second rule, the sequence is further reduced to $\kappa''\big(\kappa'(\boldsymbol{A})\big) = \kappa(\boldsymbol{A}) = \boldsymbol{Y} = (\alpha, \alpha, \beta)$.

To further clarify the two-rule conversion, let us consider a more general example, where we have the alphabet $\mathbb{A} \coloneqq \{\alpha, \beta\}$. The left column in Table B.1 shows all possible label sequences for $M \leq 2$. All possible corresponding alignments of length $N = 4$ are shown in the right columns of Table B.1.

Given an alignment $\boldsymbol{A}$ and the sequence of probability distributions, we can compute the probability

$$P(\boldsymbol{A}|\boldsymbol{X}) = \prod_{n=1}^{N} p_n(\boldsymbol{a}_n) \tag{B.7}$$

of the alignment. When computing the CTC loss, we do not explicitly know the alignment, but we know the label sequence. Because a label sequence $\boldsymbol{Y}$ can correspond to multiple alignments (see Table B.1), we have to consider all alignments that satisfy $\kappa(\boldsymbol{A}) = \boldsymbol{Y}$. Let us denote $\mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}} = \{\boldsymbol{A} \in (\mathbb{A}')^N : \kappa(\boldsymbol{A}) = \boldsymbol{Y}\}$ the set of all possible alignments of length N that can be reduced to a given label sequence $\boldsymbol{Y}$. The probability

Table B.1: Possible label sequences $\boldsymbol{Y}$ and corresponding alignments $\boldsymbol{A}$ for $\mathbb{A} = \{\alpha, \beta\}$, $M \leq 2$, and $N = 4$.

$\boldsymbol{Y}$	$\boldsymbol{A}$		
(α)	$(\alpha, \alpha, \alpha, \alpha)$	$(\alpha, \alpha, \alpha, \epsilon)$	$(\alpha, \alpha, \epsilon, \epsilon)$
	$(\alpha, \epsilon, \epsilon, \epsilon)$	$(\epsilon, \alpha, \alpha, \alpha)$	$(\epsilon, \alpha, \alpha, \epsilon)$
	$(\epsilon, \alpha, \epsilon, \epsilon)$	$(\epsilon, \epsilon, \alpha, \alpha)$	$(\epsilon, \epsilon, \alpha, \epsilon)$
	$(\epsilon, \epsilon, \epsilon, \alpha)$		
(β)	$(\beta, \beta, \beta, \beta)$	$(\beta, \beta, \beta, \epsilon)$	$(\beta, \beta, \epsilon, \epsilon)$
	$(\beta, \epsilon, \epsilon, \epsilon)$	$(\epsilon, \beta, \beta, \beta)$	$(\epsilon, \beta, \beta, \epsilon)$
	$(\epsilon, \beta, \epsilon, \epsilon)$	$(\epsilon, \epsilon, \beta, \beta)$	$(\epsilon, \epsilon, \beta, \epsilon)$
	$(\epsilon, \epsilon, \epsilon, \beta)$		
(α, α)	$(\alpha, \alpha, \epsilon, \alpha)$	$(\alpha, \epsilon, \alpha, \alpha)$	$(\alpha, \epsilon, \alpha, \epsilon)$
	$(\alpha, \epsilon, \epsilon, \alpha)$	$(\epsilon, \alpha, \epsilon, \alpha)$	
(α, β)	$(\alpha, \alpha, \alpha, \beta)$	$(\alpha, \alpha, \beta, \beta)$	$(\alpha, \alpha, \beta, \epsilon)$
	$(\alpha, \alpha, \epsilon, \beta)$	$(\alpha, \beta, \beta, \beta)$	$(\alpha, \beta, \beta, \epsilon)$
	$(\alpha, \beta, \epsilon, \epsilon)$	$(\alpha, \epsilon, \beta, \beta)$	$(\alpha, \epsilon, \beta, \epsilon)$
	$(\alpha, \epsilon, \epsilon, \beta)$	$(\epsilon, \alpha, \alpha, \beta)$	$(\epsilon, \alpha, \beta, \beta)$
	$(\epsilon, \alpha, \beta, \epsilon)$	$(\epsilon, \alpha, \epsilon, \beta)$	$(\epsilon, \epsilon, \alpha, \beta)$
(β, α)	$(\beta, \alpha, \alpha, \alpha)$	$(\beta, \alpha, \alpha, \epsilon)$	$(\beta, \alpha, \epsilon, \epsilon)$
	$(\beta, \beta, \alpha, \alpha)$	$(\beta, \beta, \alpha, \epsilon)$	$(\beta, \beta, \beta, \alpha)$
	$(\beta, \beta, \epsilon, \alpha)$	$(\beta, \epsilon, \alpha, \alpha)$	$(\beta, \epsilon, \alpha, \epsilon)$
	$(\beta, \epsilon, \epsilon, \alpha)$	$(\epsilon, \beta, \alpha, \alpha)$	$(\epsilon, \beta, \alpha, \epsilon)$
	$(\epsilon, \beta, \beta, \alpha)$	$(\epsilon, \beta, \epsilon, \alpha)$	$(\epsilon, \epsilon, \beta, \alpha)$
(β, β)	$(\beta, \beta, \epsilon, \beta)$	$(\beta, \epsilon, \beta, \beta)$	$(\beta, \epsilon, \beta, \epsilon)$
	$(\beta, \epsilon, \epsilon, \beta)$	$(\epsilon, \beta, \epsilon, \beta)$	

of the label sequence can be computed by summing the probabilities of all corresponding alignments.

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{\boldsymbol{A} \in \mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}}} P(\boldsymbol{A}|\boldsymbol{X}). \tag{B.8}$$

The final CTC loss for a single training pair is

$$L_\theta(\boldsymbol{X}, \boldsymbol{Y}) = -\log P(\boldsymbol{Y}|\boldsymbol{X}). \tag{B.9}$$

This loss function is used in mini-batch gradient descent to update the parameters θ by averaging the loss value over all training pairs in a mini batch. By this procedure, the network's parameters improve to produce probability sequences that make the ground-truth label sequences of the training set more probable. The algorithm in Table B.2 summarizes the computation of the CTC loss.

B.3 Dynamic Programming

Depending on N, we have a combinatorial explosion in the cardinality of the set $\mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}}$. Therefore, the naive approach to compute the CTC loss is infeasible for larger N. Graves et al. [71] described how to

1:	**function** NaiveCtc($\boldsymbol{X}, \boldsymbol{Y}, f_\theta$)
2:	**compute** $\mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}}$ ▷ Variables
3:	$(p_1, p_2, \ldots, p_N) \leftarrow f_\theta(\boldsymbol{X})$
4:	$P(\boldsymbol{Y}\|\boldsymbol{X}) \leftarrow 0$ ▷ Computation
5:	**for** $\boldsymbol{A} \in \mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}}$ **do**
6:	$P(\boldsymbol{A}\|\boldsymbol{X}) \leftarrow 1$
7:	**for** $n \in [1 : N]$ **do**
8:	$P(\boldsymbol{A}\|\boldsymbol{X}) \leftarrow P(\boldsymbol{A}\|\boldsymbol{X}) \cdot p_n(\boldsymbol{a}_n)$
9:	**end for**
10:	$P(\boldsymbol{Y}\|\boldsymbol{X}) \leftarrow P(\boldsymbol{Y}\|\boldsymbol{X}) + P(\boldsymbol{A}\|\boldsymbol{X})$
11:	**end for**
12:	**return** $-\log P(\boldsymbol{Y}\|\boldsymbol{X})$ ▷ Return
13:	**end function**

Table B.2: Naive approach to compute the CTC loss.

compute $P(\boldsymbol{Y}|\boldsymbol{X})$ in a differentiable and efficient way using dynamic programming similar to the forward algorithm for HMMs [76, 150].

We first define a modified label sequence

$$\boldsymbol{Z} = (z_1, z_2, \ldots, z_{2M+1}), \tag{B.10}$$

which corresponds to the label sequence $\boldsymbol{Y}$, but with the blank symbol ϵ added at the beginning, the end, and between each two consecutive elements of $\boldsymbol{Y}$. Thus, the length of $\boldsymbol{Z}$ is $2M + 1$. For example, if $\boldsymbol{Y} = (\alpha, \alpha, \beta)$, then $\boldsymbol{Z} = (\epsilon, \alpha, \epsilon, \alpha, \epsilon, \beta, \epsilon)$.

Let us now introduce a prefix notation. $\boldsymbol{Z}(1 : m)$ denotes a prefix of $\boldsymbol{Z}$ consisting of the first m elements. $\boldsymbol{A}(1 : n)$ denotes a prefix of $\boldsymbol{A}$ with the first n elements. Furthermore, $\mathbb{K}'_{\boldsymbol{X},\boldsymbol{Z}} = \{\boldsymbol{A} \in (\mathbb{A}')^N : \kappa'(\boldsymbol{A}) = \boldsymbol{Z}\}$ denotes the set of all alignments of length N that map to the modified label sequence. Then, we define a forward matrix

$$\boldsymbol{D}(m, n) \coloneqq \sum_{\substack{\boldsymbol{A} \in \mathbb{K}'_{\boldsymbol{X},\boldsymbol{Z}} \\ \kappa'(\boldsymbol{A}(1:n)) = \boldsymbol{Z}(1:m)}} \prod_{i=1}^{n} p_i(\boldsymbol{a}_i) \tag{B.11}$$

for $m \in [1 : 2M + 1]$ and $n \in [1 : N]$. In other words, $\boldsymbol{D}(m, n)$ is the probability that the first n elements in the alignment $\boldsymbol{A}$ correspond to the first m symbols in the modified label sequence $\boldsymbol{Z}$. Consequently, $\boldsymbol{D}(2M, N)$ is the probability that the alignment fully corresponds to the modified label sequence and ends with the last symbol of the label sequence. Similarly, $\boldsymbol{D}(2M + 1, N)$ is the probability that the modified

function DynamicCtc($\boldsymbol{X}, \boldsymbol{Y}, f_\theta$)

initialize $\boldsymbol{D}$ ▷ (Variables)
compute $\boldsymbol{Z}$
$(p_1, p_2, \ldots, p_N) \leftarrow f_\theta(\boldsymbol{X})$

$\boldsymbol{D}(1,1) \leftarrow p_1(\epsilon)$ ▷ (Initialization)
$\boldsymbol{D}(2,1) \leftarrow p_1(\boldsymbol{y}_1)$
for $m \in [3 : 2M+1]$ **do**
    $\boldsymbol{D}(m,1) \leftarrow 0$
end for

for $n \in [2 : N]$ **do** ▷ (Recursion)
    for $m \in [1 : 2M+1]$ **do**
        if $z_m = \epsilon$ **or** $z_{m-2} = z_m$ **then**
            $\boldsymbol{D}(m,n) \leftarrow p_n(z_m)\big(\boldsymbol{D}(m,n-1) + \boldsymbol{D}(m-1,n-1)\big)$
        else
            $\boldsymbol{D}(m,n) \leftarrow p_n(z_m)\big(\boldsymbol{D}(m,n-1) + \boldsymbol{D}(m-1,n-1) + \boldsymbol{D}(m-2,n-1)\big)$
        end if
    end for
end for

return $-\log\big(\boldsymbol{D}(2M+1,N) + \boldsymbol{D}(2M,N)\big)$ ▷ (Return)

end function

Table B.3: Dynamic-programming approach to compute the CTC loss.

label sequence ends with the blank symbol ϵ. Therefore, the final probability needed for the CTC loss is

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \boldsymbol{D}(2M+1,N) + \boldsymbol{D}(2M,N). \tag{B.12}$$

We can recursively compute the matrix $\boldsymbol{D}$. First, we initialize the first row of the matrix.

$$\boldsymbol{D}(1,1) = p_1(z_1) = p_1(\epsilon) \tag{B.13}$$

$$\boldsymbol{D}(2,1) = p_1(z_2) = p_1(\boldsymbol{y}_1) \tag{B.14}$$

$$\boldsymbol{D}(m,1) = 0, \qquad \forall m > 2. \tag{B.15}$$

In other words, the alignment either starts with the blank symbol ϵ or with the first symbol from the label sequence.

Then, we fill the matrix recursively for $n > 1$.

$$\boldsymbol{D}(m,n) = p_n(z_m) \cdot \begin{cases} \big(\boldsymbol{D}(m,n-1) + \boldsymbol{D}(m-1,n-1)\big), & \text{if } z_m = \epsilon \text{ or } z_{m-2} = z_m, \\ \big(\boldsymbol{D}(m,n-1) + \boldsymbol{D}(m-1,n-1) + \boldsymbol{D}(m-2,n-1)\big), & \text{otherwise.} \end{cases} \tag{B.16}$$

This recursion has a differentiation of three different cases for finding out at each time step which symbols are possible at the previous time step. As the first case ($z_m = \epsilon$), we have the blank symbol ϵ at the current time step n. The symbol of the last time step $n-1$ can be either a blank symbol $z_m = \epsilon$ (no symbol change) or the previous symbol z_{m-1} of the modified label sequence. As the second case ($z_{m-2} = z_m$), at the current time step n, we cover a symbol z_m from the modified label sequence, which is identical to its pre-predecessor z_{m-2}. The in-between symbol z_{m-1} is a blank ϵ (as always in the modified label sequence). As a consequence, the symbol of the last time step $n-1$ can be either z_m (no symbol change) or $z_{m-1} = \epsilon$. It is not allowed to have z_{m-2} as a previous symbol because the blank symbol ϵ has to be used to indicate consecutive duplicate symbols in the label sequence. As the third case ("otherwise"), the previous symbol can be either z_m (no symbol change), or z_{m-1} (blank symbol ϵ), or z_{m-2} (symbol change without using an intermediate ϵ).

The algorithm in Table B.3 summarizes the dynamic-programming approach to compute the CTC loss.

B.4 Toy Example

We further illustrate the naive and the dynamic-programming approach to compute the CTC loss with a toy example. Let our modified alphabet be $\mathbb{A}' = \{\alpha, \beta, \epsilon\}$. We have a short label sequence of $\boldsymbol{Y} = (\alpha, \beta)$. Thus, the modified label sequence is $\boldsymbol{Z} = (\epsilon, \alpha, \epsilon, \beta, \epsilon)$. Furthermore, we fix $N = 3$. Interpreting the sequence of probability distributions $f_\theta(\boldsymbol{X}) = (p_1, p_2, p_3)$ as a matrix, we set it to

$$\begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{pmatrix} 0.4 & 0.0 & 0.0 \\ 0.2 & 0.5 & 0.5 \\ 0.4 & 0.5 & 0.5 \end{pmatrix} \begin{matrix} \alpha \\ \beta \\ \epsilon \end{matrix} \end{array} . \tag{B.17}$$

Let us first take the naive approach. For this short example, we can write down all alignments and compute their probabilities.

$$\boldsymbol{A} = (\alpha, \alpha, \beta) \qquad P(\boldsymbol{A}|\boldsymbol{X}) = 0.4 \cdot 0.0 \cdot 0.5 = 0.0 \tag{B.18}$$

$$\boldsymbol{A} = (\alpha, \beta, \beta) \qquad P(\boldsymbol{A}|\boldsymbol{X}) = 0.4 \cdot 0.5 \cdot 0.5 = 0.1 \tag{B.19}$$

$$\boldsymbol{A} = (\alpha, \beta, \epsilon) \qquad P(\boldsymbol{A}|\boldsymbol{X}) = 0.4 \cdot 0.5 \cdot 0.5 = 0.1 \tag{B.20}$$

$$\boldsymbol{A} = (\alpha, \epsilon, \beta) \qquad P(\boldsymbol{A}|\boldsymbol{X}) = 0.4 \cdot 0.5 \cdot 0.5 = 0.1 \tag{B.21}$$

$$\boldsymbol{A} = (\epsilon, \alpha, \beta) \qquad P(\boldsymbol{A}|\boldsymbol{X}) = 0.4 \cdot 0.0 \cdot 0.5 = 0.0 \tag{B.22}$$

If we sum the probabilities for the alignments, we get our final probability for the label sequence: $P(\boldsymbol{Y}|\boldsymbol{X}) = 0.3$. We can also compute this value with the help of the forward matrix $\boldsymbol{D}$.

$$\boldsymbol{D}^{\top} = \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0.4 & 0.2 & 0.1 \\ 0.4 & 0.0 & 0.0 \\ 0.0 & 0.2 & 0.1 \\ 0.0 & 0.2 & 0.2 \\ 0.0 & 0.0 & 0.1 \end{pmatrix} \end{array} \begin{array}{l} z_1{=}\epsilon \\ z_2{=}\alpha \\ z_3{=}\epsilon \\ z_4{=}\beta \\ z_5{=}\epsilon \end{array} \tag{B.23}$$

In this matrix, we see that $P(\boldsymbol{Y}|\boldsymbol{X}) = \boldsymbol{D}(2M+1, N) + \boldsymbol{D}(2M, N) = 0.1 + 0.2 = 0.3$. We achieved the same result by both the naive as well as the dynamic-programming approach. For this example, the CTC loss is $L_\theta(\boldsymbol{X}, \boldsymbol{Y}) = -\log 0.3$.

B.5 Musical Example

In Chapter 7, we applied the CTC loss in a musical scenario. We conclude this section by illustrating the explained concepts using a musical example. Let us consider $\boldsymbol{X}$ to be a sequence of spectral vectors computed from a music recording and $\boldsymbol{Y}$ to be a sequence of chroma labels. Figure B.1 shows such a pair using the beginning of a central theme from Beethoven's *Great Fugue* Op. 133. In this figure, we also display the graphical sheet music corresponding to that music excerpt. We have a sequence of five notes, each interrupted by rests. The figure also shows the chroma label sequence $\boldsymbol{Y}$ and the modified label sequence $\boldsymbol{Z}$. Note that Beethoven's unusual notation (tied eighth notes) is not important for our discussion. The figure shows the network's input representation $\boldsymbol{X}$ of length $N = 11$ on the upper left and output $f_\theta(\boldsymbol{X})$ of the network on the upper right (using logarithmic compression for graphical reasons). Even though the ϵ symbol has the highest probability for all time frames, we can clearly follow the theme's chroma sequence in the network's output. The set $\mathbb{K}_{\boldsymbol{X},\boldsymbol{Y}}$ contains all possible alignments of length N that map to the label sequence $\boldsymbol{Y}$. In our specific example, there are more than three thousand valid alignments (satisfying $\kappa(\boldsymbol{A}) = \boldsymbol{Y}$). The figure only illustrates three elements from this set. For a given alignment, red rectangles indicate the selected symbol for a given time step. The first note $\mathrm{B}^\flat$ of the theme is repeated. In any valid alignment, two $\mathrm{B}^\flat$ symbols need to be interrupted by at least one ϵ symbol. The middle alignment of Figure B.1 goes especially well with the network output. Computing the probability for an alignment consists of multiplying the probabilities for the activate symbol of each time step (red rectangles). Computing the CTC loss then involves adding the probabilities for all three thousand valid alignments.

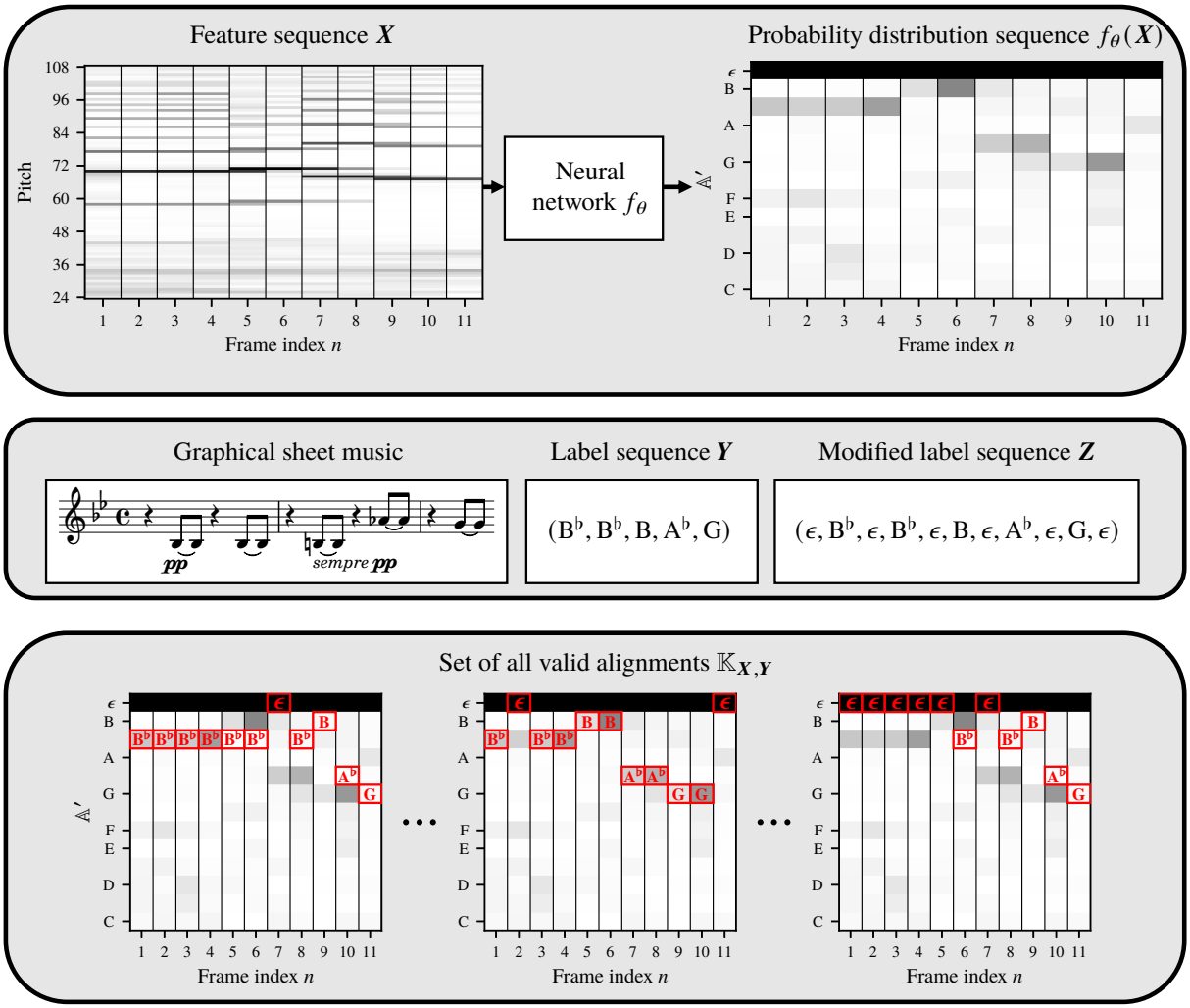


Figure B.1: Illustration for an example using the beginning of a central theme from Beethoven's *Great Fugue* Op. 133.

Abbreviations

AMT	automatic music transcription
BM	Barlow & Morgenstern [12]
CENS	chroma energy distribution normalized statistics
CNN	convolutional neural network
CQT	constant-Q transform
CTC	connectionist temporal classification
DFT	discrete Fourier transform
DNN	deep neural network
EDM	electronic dictionary of musical themes
FFT	fast Fourier transform
FIR	finite impulse response
HCQT	harmonic constant-Q transform
HMM	hidden Markov model
HNSW	hierarchical navigable small world
IIR	infinite impulse response
LSH	locality-sensitive hashing
MAP	mean average precision
MIDI	musical instrument digital interface
MIR	music information retrieval
MRR	mean reciprocal rank
MTD	musical theme dataset
OCR	optical character recognition
OMR	optical music recognition
PCA	principal component analysis
SDTW	subsequence dynamic time warping
STFT	short-time Fourier fransform

Bibliography

[1] Hadrien Foroughmand Aarabi and Geoffroy Peeters. Deep-rhythm for global tempo estimation in music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 636–643, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527890.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, Savannah, Georgia, USA, 2016.

[3] Tobi Adewoye, Xiao Han, Nick Ruest, Ian Milligan, Samantha Fritz, and Jimmy Lin. Content-based exploration of archival images using neural networks. In *Proceedings of the Joint Conference on Digital Libraries (JCDL)*, pages 489–490, Virtual Event, Wuhan, China, 2020. doi: 10.1145/3383583.3398577.

[4] Ruchit Agrawal and Simon Dixon. Learning frame similarity using siamese networks for audio-to-score alignment. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 141–145, Amsterdam, The Netherlands, 2020. doi: 10.23919/Eusipco47968.2020.9287625.

[5] Blaise Agüera y Arcas, Beat Gfeller, Ruiqi Guo, Kevin Kilgour, Sanjiv Kumar, James Lyon, Julian Odell, Marvin Ritter, Dominik Roblek, Matthew Sharifi, and Mihajlo Velimirović. Now playing: Continuous low-power music recognition. *CoRR*, abs/1711.10958, 2017.

[6] Andreas Arzt and Stefan Lattner. Audio-to-score alignment using transposition-invariant features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 592–599, Paris, France, 2018. doi: 10.5281/zenodo.1492485.

[7] Andreas Arzt and Gerhard Widmer. Piece identification in classical piano music without reference scores. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 354–360, Suzhou, China, 2017. doi: 10.5281/zenodo.1417673.

[8] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–438, Porto, Portugal, 2012. doi: 10.5281/zenodo.1417022.

[9] Stefan Balke, Sanu Pulimootil Achankunju, and Meinard Müller. Matching musical themes based on noisy OCR and OMR input. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 703–707, Brisbane, Australia, 2015. doi: 10.1109/ICASSP.2015.7178060.

[10] Stefan Balke, Vlora Arifi-Müller, Lukas Lamprecht, and Meinard Müller. Retrieving audio recordings using musical themes. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 281–285, Shanghai, China, 2016. doi: 10.1109/ICASSP.2016.7471681.

[11] Stefan Balke, Christian Dittmar, Jakob Abeßer, and Meinard Müller. Data-driven solo voice enhancement for jazz music retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 196–200, New Orleans, Louisiana, USA, 2017. doi: 10.1109/ICASSP.2017.7952145.

[12] Harold Barlow and Sam Morgenstern. *A Dictionary of Musical Themes*. Crown Publishers, Inc., revised edition third printing edition, 1975.

[13] Harold Barlow and Sam Morgenstern. *A Dictionary of Opera and Song Themes*. Crown Publishers, Inc., revised edition edition, 1976.

[14] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005. doi: 10.1109/TMM.2004.840597.

[15] Dogac Basaran, Slim Essid, and Geoffroy Peeters. Main melody estimation with source-filter NMF and CRNN. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 82–89, Paris, France, 2018. doi: 10.5281/zenodo.1492349.

[16] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019. doi: 10.1109/MSP.2018.2869928.

[17] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi: 10.1109/TPAMI.2013.50.

[18] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. doi: 10.1145/361002.361007.

[19] Bruce Benward and Marilyn Saker. *Music in Theory and Practice*. McGraw Hill, 8th edition, 2009.

[20] Tamar Berman, J. Stephen Downie, and Bart Berman. Beyond error tolerance: Finding thematic similarities in music digital libraries. In *Proceedings of the European Conference on Digital Libraries (ECDL)*, pages 463–466, Alicante, Spain, 2006. doi: 10.1007/11863878_44.

[21] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, Taipei, Taiwan, 2014. doi: 10.5281/zenodo.1417889.

[23] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan P. Bello. Deep salience representations for F0 tracking in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 63–70, Suzhou, China, 2017. doi: 10.5281/zenodo.1417937.

[24] Benjamin Blankertz. The constant Q transform. Technical report, University of Münster.

[25] Juan J. Bosch and Emilia Gómez. Melody extraction for MIREX 2016. In *Music Information Retrieval Evaluation eXchange (MIREX) System Abstracts*. 2016.

[26] Juan J. Bosch and Emilia Gómez. Melody extraction based on a source-filter model using pitch contour selection. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 67–74, Hamburg, Germany, 2016. doi: 10.5281/zenodo.851187.

[27] Juan J. Bosch, Ricard Marxer, and Emilia Gómez. Evaluation and combination of pitch estimation methods for melody extraction in symphonic classical music. *Journal of New Music Research*, 45(2):101–117, 2016. doi: 10.1080/09298215.2016.1182191.

[28] Julian Brandner. Efficient cross-version music retrieval using dimensionality reduction and indexing techniques. Master Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2019.

[29] Judith C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991. doi: 10.1121/1.400476.

[30] Emanuele Di Buccio, Nicola Montecchio, and Nicola Orio. A scalable cover identification engine. In *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, pages 1143–1146, Firenze, Italy, 2010. doi: 10.1145/1873951.1874171.

[31] Benjamin Bustos and Nelson Morales. On the asymptotic behavior of nearest neighbor search using pivot-based indexes. In *Proceedings of the International Workshop on Similarity Search and Applications (SISAP)*, pages 33–39, Istanbul, Turkey, 2010. doi: 10.1145/1862344.1862350.

[32] Benjamin Bustos, Gonzalo Navarro, and Edgar Chávez. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 24(14):2357–2366, 2003. doi: 10.1016/S0167-8655(03)00065-5.

[33] Donald Byrd and Jakob G. Simonsen. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, 44(3):169–195, 2015. doi: 10.1080/09298215.2015.1045424.

[34] Jorge Calvo-Zaragoza and David Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4), 2018. doi: 10.3390/app8040606.

[35] Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Antonio Pertusa. End-to-end optical music recognition using neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 472–477, Suzhou, China, 2017. doi: 10.5281/zenodo.1418333.

[36] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha. Understanding optical music recognition. *ACM Computing Surveys*, 53(4), 2020. doi: 10.1145/3397499.

[37] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of algorithms for audio fingerprinting. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 169–173, St. Thomas, Virgin Islands, USA, 2002. doi: 10.1109/MMSP.2002.1203274.

[38] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41(3):271–284, 2005. doi: 10.1007/s11265-005-4151-3.

[39] Michael A. Casey, Christophe Rhodes, and Malcolm Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5): 1015–1028, 2008. doi: 10.1109/TASL.2008.925883.

[40] Michael A. Casey, Remco Veltkap, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008. doi: 10.1109/JPROC.2008.916370.

[41] Vijay Chandrasekhar, Matt Sharifi, and David A. Ross. Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 801–806, Miami, Florida, USA, 2011. doi: 10.5281/zenodo.1415260.

[42] Sungkyun Chang, Juheon Lee, Sang Keun Choe, and Kyogu Lee. Audio cover song identification using convolutional neural network. *CoRR*, abs/1712.00166, 2017.

[43] Mark Changizi. *Harnessed: How Language and Music Mimicked Nature and Transformed Ape to Man*. BenBella Books, 2011.

[44] Wenqin Chen, Jessica Keast, Jordan Moody, Corinne Moriarty, Felicia Villalobos, Virtue Winter, Xueqi Zhang, Xuanqi Lyu, Elizabeth Freeman, Jessie Wang, Sherry Cai, and Katherine M. Kinnaird. Data usage in MIR: history & future recommendations. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 25–32, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527733.

[45] Helena Cuesta, Brian McFee, and Emilia Gómez. Multiple F0 estimation in vocal ensembles using convolutional neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 302–309, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245434.

[46] Jürgen Diet and Magda Gerritsen. Encoding, searching, and displaying of music incipits in the RISM-OPAC. In *Proceedings of the Music Encoding Conference (MEC)*, pages 11–14, Mainz, Germany, 2013.

[47] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001. doi: 10.1076/jnmr.30.1.39.7119.

[48] Guillaume Doras and Geoffroy Peeters. Cover detection using dominant melody embeddings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 107–114, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527752.

[49] Guillaume Doras and Geoffroy Peeters. A prototypical triplet loss for cover detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3797–3801, Barcelona, Spain, 2020. doi: 10.1109/ICASSP40776.2020.9054619.

[50] Guillaume Doras, Furkan Yesiler, Joan Serrà, Emilia Gómez, and Geoffroy Peeters. Combining musical features for cover detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 279–286, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245424.

[51] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 789–795, New York City, New York, USA, 2016. doi: 10.5281/zenodo.1415548.

[52] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Learning audio-sheet music correspondences for score identification and offline alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 115–122, Suzhou, China, 2017. doi: 10.5281/zenodo.1417807.

[53] Matthias Dorfer, Jan Hajič Jr., Andreas Arzt, Harald Frostel, and Gerhard Widmer. Learning audio-sheet music correspondences for cross-modal retrieval and piece identification. *Transactions of the International Society for Music Information (TISMIR)*, 1(1):22–31, 2018. doi: 10.5334/tismir.12.

[54] William Drabkin. Theme. In *Grove Music Online*. Oxford University Press, 2001. URL https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000027789.

[55] Jean-Louis Durrieu, Bertrand David, and Gaël Richard. A musically motivated mid-level representation for pitch estimation and musical audio source separation. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1180–1191, 2011. doi: 10.1109/JSTSP.2011.2158801.

[56] Daniel P.W. Ellis and Graham E. Poliner. 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)*, pages 1429–1432, Honolulu, Hawaii, USA, 2007. doi: 10.1109/ICASSP.2007.367348.

[57] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010. doi: 10.1109/TASL.2009.2038819.

[58] Slim Essid, Gaël Richard, and Bertrand David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):68–80, 2006. doi: 10.1109/TSA.2005.860351.

[59] Jiunn-Tsair Fang, Chi-Ting Day, and Pao-Chi Chang. Deep feature learning for cover song identification. *Multimedia Tools and Applications*, 76(22):23225–23238, 2017. doi: 10.1007/s11042-016-4107-6.

[60] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller. Sheet music-audio identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 645–650, Kobe, Japan, 2009. doi: 10.5281/zenodo.1416742.

[61] Jerome H. Friedman, Jon Louis Bentley, and Raphael A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software,*, 3(3):209–226, 1977. doi: 10.1145/355744.355745.

[62] Erik Frøkjær, Morten Hertzum, and Kasper Hornbæk. Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 345–352, The Hague, The Netherlands, 2000. doi: 10.1145/332040.332455.

[63] Salvador García, Joaquín Derrac, José Ramón Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(3):417–435, 2012. doi: 10.1109/TPAMI.2011.142.

[64] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 776–780, New Orleans, Louisiana, USA, 2017. doi: 10.1109/ICASSP.2017.7952261.

[65] Claudio Gentile and Manfred K. Warmuth. Linear hinge loss and average margin. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 225–231, Denver, Colorado, USA, 1998.

[66] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, pages 231–236, San Francisco, California, USA, 1995. doi: 10.1145/217279.215273.

[67] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, and Donatien Thorez. Towards modeling texture in symbolic data. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 59–64, Taipei, Taiwan, 2014. doi: 10.5281/zenodo.1415030.

[68] Emilia Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[69] Masataka Goto. Development of the RWC music database. In *Proceedings of the International Congress on Acoustics (ICA)*, pages 553–556, 2004.

[70] Prachi Govalkar, Johannes Fischer, Frank Zalkow, and Christian Dittmar. A comparison of recent neural vocoders for speech signal reconstruction. In *Proceedings of the ISCA Speech Synthesis Workshop (SSW)*, pages 7–12, Vienna, Austria, 2019. doi: 10.21437/SSW.2019-2.

[71] Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 369–376, Pittsburgh, Pennsylvania, USA, 2006. doi: 10.1145/1143844.1143891.

[72] Peter Grosche and Meinard Müller. Toward characteristic audio shingles for efficient cross-version music retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 473–476, Kyoto, Japan, 2012. doi: 10.1109/ICASSP.2012.6287919.

[73] Peter Grosche, Meinard Müller, and Joan Serrà. Audio content-based music retrieval. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 157–174. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[74] Chitralekha Gupta, Emre Yılmaz, and Haizhou Li. Automatic lyrics alignment and transcription in polyphonic music: Does background music help? In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 496–500, Barcelona, Spain, 2020. doi: 10.1109/ICASSP40776.2020.9054567.

[75] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, New York City, New York, USA, 2006. doi: 10.1109/CVPR.2006.100.

[76] Awni Hannun. *Transcribing Real-Valued Sequences with Deep Neural Network*. PhD thesis, Stanford University, 2018.

[77] John Harvill, Mohammed Abdel-Wahab, Reza Lotfian, and Carlos Busso. Retrieving speech samples with similar emotional content using a triplet loss function. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7400–7404, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8683273.

[78] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, Louisiana, USA, 2019.

[79] Hlynur D. Hlynsson, Alberto N. Escalante-B., and Laurenz Wiskott. Measuring the data efficiency of deep learning methods. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 691–698, Prague, Czech Republic, 2019.

[80] Yuanbo Hou, Qiuqiang Kong, and Shengchen Li. Audio tagging with connectionist temporal classification model using sequentially labelled data. In *Proceedings of the International Conference in Communications, Signal Processing, and Systems (CSPS)*, pages 955–964, Dalian, China, 2019. doi: 10.1007/978-981-13-6504-1_114.

[81] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, New York, USA, 2003. doi: 10.1109/ASPAA.2003.1285862.

[82] Eric J. Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 673–679, Málaga, Spain, 2015. doi: 10.5281/zenodo.1417549.

[83] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 403–408, Porto, Portugal, 2012. doi: 10.5281/zenodo.1415726.

[84] Eric J. Humphrey, Oriol Nieto, and Juan P. Bello. Data driven and discriminative projections for large-scale cover song identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 149–154, Curitiba, Brazil, 2013. doi: 10.5281/zenodo.1416548.

[85] Yun-Ning Hung and Yi-Hsuan Yang. Frame-level instrument recognition by timbre and pitch. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–142, Paris, France, 2018. doi: 10.5281/zenodo.1492363.

[86] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel P. W. Ellis, Shawn Hershey, Jiayang Liu, R. Channing Moore, and Rif A. Saurous. Unsupervised learning of semantic audio representations. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 126–130, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8461684.

[87] Chaoya Jiang, Deshun Yang, and Xiaoou Chen. Learn a robust representation for cover song identification via aggregating local and global music temporal context. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, London, UK, 2020. doi: 10.1109/ICME46284.2020.9102975.

[88] Cyril Joder, Slim Essid, and Gaël Richard. Learning optimal features for polyphonic audio-to-score alignment. *IEEE Transactions on Audio, Speech & Language Processing*, 21(10):2118–2128, 2013. doi: 10.1109/TASL.2013.2266794.

[89] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA, 2015.

[90] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[91] Stephanie Klauk and Frank Zalkow. Das italienische Streichquartett im 18. Jahrhundert. Möglichkeiten der semiautomatisierten Stilanalyse. In *Proceedings of the Jahrestagung der Gesellschaft für Musikforschung (GfM)*, Halle/Saale, Germany, 2015.

[92] Stephanie Klauk and Frank Zalkow. Methoden computergestützter melodischer Analyse am Beispiel italienischer Streichquartette. In Stephanie Klauk, editor, *Instrumentalmusik neben Haydn und Mozart. Analyse, Aufführungspraxis und Edition*, pages 151–168. Saarbrücker Studien zur Musikwissenschaft 20, Königshausen & Neumann, 2020.

[93] Andreas Kornstädt. Themefinder: A web-based melodic search tool. *Computing in Musicology*, 11, 1998.

[94] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 37–43, New York City, New York, USA, 2016. doi: 10.5281/zenodo.1416314.

[95] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Salerno, Italy, 2016. doi: 10.1109/MLSP.2016.7738895.

[96] Alexios Kotsifakos, Evangelos E. Kotsifakos, Panagiotis Papapetrou, and Vassilis Athitsos. Genre classification of symbolic music with SMBGT. In *Proceedings of the International Conference on PErvasive Technologies Related to Assistive Environments (PETRA)*, Rhodes, Greece, 2013. doi: 10.1145/2504335.2504382.

[97] Michael Krause, Frank Zalkow, Julia Zalkow, Christof Weiß, and Meinard Müller. Classifying leitmotifs in recordings of operas by Richard Wagner. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 473–480, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245472.

[98] Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, 52(2):341–378, 2017. doi: 10.1007/s10115-016-1004-2.

[99] Frank Kurth and Meinard Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, 2008. doi: 10.1109/TASL.2007.911552.

[100] Juheon Lee, Sungkyun Chang, Sang Keun Choe, and Kyogu Lee. 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)*, pages 396–400, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8461395.

[101] Kjell Lemström and Jorma Tarhio. Searching monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access – Volume 2*, RIAO '00, pages 1261–1279, Paris, France, 2000.

[102] Alexander Lerch, Claire Arthur, Ashis Pati, and Siddharth Gururani. Music performance analysis: A survey. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 33–43, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527735.

[103] Jerrold Levinson. What a musical work is. *The Journal of Philosophy*, 77(1), 1980.

[104] David B. Levy. "Ma però beschleunigend": Notation and meaning in ops. 133/134. *Beethoven Forum*, 14(2): 129–149, 2007.

[105] Mingyu Li and Ning Chen. A robust cover song identification system with two-level similarity fusion and post-processing. *Applied Sciences*, 8(8), 2018. doi: 10.3390/app8081383.

[106] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2019. doi: 10.1109/TKDE.2019.2909204.

[107] Cynthia C. S. Liem and Alan Hanjalic. Cover song retrieval: A comparative study of system component choices. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 573–578, Kobe, Japan, 2009. doi: 10.5281/zenodo.1414896.

[108] Cynthia C. S. Liem, Meinard Müller, Douglas Eck, George Tzanetakis, and Alan Hanjalic. The need for music information retrieval with user-centered and multimodal strategies. In *Proceedings of the International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies (MIRUM)*, pages 1–6, 2011. doi: 10.1145/2072529.2072531.

[109] Ting Liu, Andrew W. Moore, Alexander G. Gray, and Ke Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 825–832, Vancouver, Canada, 2004.

[110] Justin London. Building a representative corpus of classical music. *Music Perception*, 31(1):68–90, 2013. doi: 10.1525/mp.2013.31.1.68.

[111] Rui Lu, Kailun Wu, Zhiyao Duan, and Changshui Zhang. Deep ranking: Triplet MatchNet for music metric learning. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–125, New Orleans, Louisiana, USA, 2017. doi: 10.1109/ICASSP.2017.7952130.

[112] Anna Lubiw and Luke Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 289–296, Barcelona, Spain, 2004. doi: 10.5281/zenodo.1417969.

[113] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, Georgia, USA, 2013.

[114] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020. doi: 10.1109/TPAMI.2018.2889473.

[115] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[116] Rosalind B. Marimont and Marvin B. Shapiro. Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70, 1979. doi: 10.1093/imamat/24.1.59.

[117] Brian McFee and Juan Pablo Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 188–194, Suzhou, China, 2017. doi: 10.5281/zenodo.1414880.

[118] Brian McFee and Gert R. G. Lanckriet. Large-scale music similarity search with spatial trees. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 55–60, Miami, Florida, USA, 2011. doi: 10.5281/zenodo.1414930.

[119] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa: Audio and music signal analysis in Python. In *Proceedings the Python Science Conference*, pages 18–25, Austin, Texas, USA, 2015. doi: 10.25080/Majora-7b98e3ed-003.

[120] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M. Bittner, and Juan Pablo Bello. Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research. *IEEE Signal Processing Magazine*, 36(1):128–137, 2019. doi: 10.1109/MSP.2018.2875349.

[121] Colin Meek and William P. Birmingham. Thematic extractor. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Bloomington, Indiana, USA, 2001. doi: 10.5281/zenodo.1414828.

[122] Riccardo Miotto and Nicola Orio. A methodology for the segmentation and identification of music works. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 273–278, Vienna, Austria, 2007. doi: 10.5281/zenodo.1415952.

[123] Riccardo Miotto and Nicola Orio. A music identification system based on chroma indexing and statistical modeling. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 301–306, Philadelphia, Pennsylvania, USA, 2008. doi: 10.5281/zenodo.1415254.

[124] Nicola Montecchio, Emanuele Di Buccio, and Nicola Orio. An efficient identification methodology for improved access to music heritage collections. *Journal of Multimedia*, 7(2):145–158, 2012.

[125] Veronica Morfi, Yves Bas, Hanna Pamula, Hervé Glotin, and Dan Stowell. NIPS4Bplus: A richly annotated birdsong audio dataset. *PeerJ Computer Science*, 5:e223, 2019. doi: 10.7717/peerj-cs.223.

[126] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014. doi: 10.1109/TPAMI.2014.2321376.

[127] Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.

[128] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[129] Meinard Müller and Sebastian Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 215–220, Miami, Florida, USA, 2011. doi: 10.5281/zenodo.1416032.

[130] Meinard Müller and Frank Zalkow. FMP Notebooks: Educational material for teaching and learning fundamentals of music processing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 573–580, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527872.

[131] Meinard Müller, Frank Kurth, and Tido Röder. Towards an efficient algorithm for automatic score-to-audio synchronization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 365–372, Barcelona, Spain, 2004. doi: 10.5281/zenodo.1416302.

[132] Meinard Müller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Proceedings of the IEEE Workshop on Applications of Signal Processing (WASPAA)*, pages 275–278, New Paltz, New York, USA, 2005. doi: 10.1109/ASPAA.2005.1540223.

[133] Meinard Müller, Verena Konz, Andi Scharfstein, Sebastian Ewert, and Michael Clausen. Towards automated extraction of tempo parameters from expressive music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 69–74, Kobe, Japan, 2009. doi: 10.5281/zenodo.1416024.

[134] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlora Arifi-Müller. Saarland music data (SMD). In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, USA, 2011.

[135] Meinard Müller, Helmut Hedwig, Frank Zalkow, and Stefan Popescu. Constraint-based time-scale modification of music recordings for noise beautification. *Applied Sciences*, 8(3), 2018. doi: 10.3390/app8030436.

[136] Meinard Müller, Andreas Arzt, Stefan Balke, Matthias Dorfer, and Gerhard Widmer. Cross-modal music retrieval and applications: An overview of key methodologies. *IEEE Signal Processing Magazine*, 36(1): 52–62, 2019. doi: 10.1109/MSP.2018.2868887.

[137] Ryo Nishikimi, Eita Nakamura, Satoru Fukayama, Masataka Goto, and Kazuyoshi Yoshii. Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 161–165, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8683024.

[138] Julien Osmalskyj, Marc Van Droogenbroeck, and Jean-Jacques Embrechts. Enhancing cover song identification with hierarchical rank aggregation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 136–142, New York City, New York, USA, 2016. doi: 10.5281/zenodo.1418109.

[139] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam. Representation learning of music using artist labels. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 717–724, Paris, France, 2018. doi: 10.5281/zenodo.1492517.

[140] Denys Parsons. *The Directory of Tunes and Musical Themes*. S. Brown, 1975.

[141] Johan Pauwels, Ken O'Hanlon, Emilia Gómez, and Mark B. Sandler. 20 years of automatic chord recognition from audio. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 54–63, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527739.

[142] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[143] Jeremy Pickens, Juan Pablo Bello, Tim Crawford, Matthew J. Dovey, Giuliano Monti, and Mark B. Sandler. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2002. doi: 10.5281/zenodo.1418091.

[144] Emanuele Pollastri and Giuliano Simoncelli. Classification of melodies by composer with hidden markov models. In *Proceedings of the International Conference on WEB Delivering of Music (WEDELMUSIC)*, pages 88–95, Florence, Italy, 2001. doi: 10.1109/WDM.2001.990162.

[145] Lutz Prechelt and Rainer Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8(2):133–149, 2001. doi: 10.1145/376929.376978.

[146] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing*. Prentice Hall, 1996.

[147] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. Graph-based nearest neighbor search: From practice to theory. In *Proceedings of the International Conference on Machine Learning (ICML)*, Vienna, Austria, 2020.

[148] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara N. Sainath. Deep learning for audio signal processing. *IEEE Journal on Selected Topics in Signal Processing*, 13(2):206–219, 2019. doi: 10.1109/JSTSP.2019.2908700.

[149] Xiaoyu Qi, Deshun Yang, and Xiaoou Chen. Triplet convolutional network for music version identification. In *Proceedings of the International Conference on Multimedia Modeling (MMM)*, pages 544–555, Bangkok, Thailand, 2018. doi: 10.1007/978-3-319-73603-7_44.

[150] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

[151] Colin Raffel and Daniel P. W. Ellis. Intuitive analysis, creation and manipulation of MIDI data with pretty_midi. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014.

[152] Colin Raffel and Daniel P. W. Ellis. Large-scale content-based matching of MIDI and audio files. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 234–240, Málaga, Spain, 2015. doi: 10.5281/zenodo.1417371.

[153] Colin Raffel and Daniel P. W. Ellis. Optimizing DTW-based audio-to-MIDI alignment and matching. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 81–85, Shanghai, China, 2016. doi: 10.1109/ICASSP.2016.7471641.

[154] Zafar Rafii, Bob Coover, and Jinyu Han. An audio fingerprinting system for live version identification using image processing techniques. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 644–648, Florence, Italy, 2014. doi: 10.1109/ICASSP.2014.6853675.

[155] Parikshit Ram and Kaushik Sinha. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1378–1388, Anchorage, Alaska, USA, 2019. doi: 10.1145/3292500.3330875.

[156] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: State-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012. doi: 10.1007/s13735-012-0004-6.

[157] Josh Reiss, Jean-Julien Aucouturier, and Mark Sandler. Efficient multidimensional searching routines. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Bloomington, Indiana, USA, 2001. doi: 10.5281/zenodo.1415546.

[158] Rudolph Reti. *The Thematic Process in Music*. The Macmillan Company, 1951.

[159] Juan Ramón Rico-Juan, Jose J. Valero-Mas, and Jorge Calvo-Zaragoza. Extensions to rank-based prototype selection in k-nearest neighbour classification. *Applied Soft Computing*, 85, 2019. doi: 10.1016/j.asoc.2019.105803.

[160] Miguel A. Román, Antonio Pertusa, and Jorge Calvo-Zaragoza. An end-to-end framework for audio-to-score music transcription on monophonic excerpts. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 34–41, Paris, France, 2018. doi: 10.5281/zenodo.1492337.

[161] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI*, pages 234–241, Munich, Germany, 2015. doi: 10.1007/978-3-319-24574-4_28.

[162] Sebastian Rosenzweig, Frank Scherbaum, David Shugliashvili, Vlora Arifi-Müller, and Meinard Müller. Erkomaishvili Dataset: A curated corpus of traditional Georgian vocal music for computational musicology. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1):31–41, 2020. doi: 10.5334/tismir.44.

[163] Jimena Royo-Letelier, Romain Hennequin, Viet-Anh Tran, and Manuel Moussallam. Disambiguating music artists at scale with audio metric learning. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 622–629, Paris, France, 2018. doi: 10.5281/zenodo.1492493.

[164] Matti Ryynänen and Anssi Klapuri. Query by humming of MIDI and audio using locality sensitive hashing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2249–2252, Las Vegas, Nevada, USA, 2008. doi: 10.1109/ICASSP.2008.4518093.

[165] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012. doi: 10.1109/TASL.2012.2188515.

[166] Justin Salamon, Joan Serrà, and Emilia Gómez. Tonal representations for music retrieval: From version identification to query-by-humming. *International Journal of Multimedia Information Retrieval*, 2(1):45–58, 2013. doi: 10.1007/s13735-012-0026-0.

[167] Justin Salamon, Emilia Gómez, Daniel P. W. Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014. doi: 10.1109/MSP.2013.2271648.

[168] Craig Stuart Sapp. Comparative analysis of multiple musical performances. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 497–500, Vienna, Austria, 2007. doi: 10.5281/zenodo.1417693.

[169] Marc Sarfati, Anthony Hu, and Jonathan Donier. Community-based cover song detection. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 244–250, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527790.

[170] Jan Schlüter. Learning binary codes for efficient large-scale music similarity search. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 581–586, Curitiba, Brazil, 2013. doi: 10.5281/zenodo.1416508.

[171] Christian Schörkhuber and Anssi P. Klapuri. Constant-Q transform toolbox for music processing. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010. doi: 10.5281/zenodo.849741.

[172] Hendrik Schreiber, Frank Zalkow, and Meinard Müller. Modeling and estimating local tempo: A case study on Chopin's mazurkas. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 773–779, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245546.

[173] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, Boston, Massachusetts, USA, 2015. doi: 10.1109/CVPR.2015.7298682.

[174] Prem Seetharaman and Zafar Rafii. Cover song identification with 2D fourier transform sequences. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 616–620, New Orleans, Louisiana, USA, 2017. doi: 10.1109/ICASSP.2017.7952229.

[175] Joan Serrà, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16: 1138–1151, 2008. doi: 10.1109/TASL.2008.924595.

[176] Joan Serrà, Xavier Serra, and Ralph G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9), 2009. doi: 10.1088/1367-2630/11/9/093017.

[177] Joan Serrà, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: Background, approaches, evaluation and beyond. In Z. W. Ras and A. A. Wieczorkowska, editors, *Advances in Music Information Retrieval*, volume 274 of *Studies in Computational Intelligence*, chapter 14, pages 307–332. Springer, Berlin, Germany, 2010. doi: 10.1007/978-3-642-11674-2_14.

[178] Joan Serrà, Massimiliano Zanin, Perfecto Herrera, and Xavier Serra. Characterization and exploitation of community structure in cover song networks. *Pattern Recognition Letters*, 33(9):1032–1041, 2012. doi: 10.1016/j.patrec.2012.02.013.

[179] Xavier Serra. Creating research corpora for the computational study of music: The case of the CompMusic project. In *Proceedings of the AES International Conference on Semantic Audio*, London, UK, 2014.

[180] Federico Simonetta, Carlos Eduardo Cancino Chacón, Stavros Ntalampiras, and Gerhard Widmer. A convolutional approach to melody line identification in symbolic scores. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 924–931, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527966.

[181] Dean Keith Simonton. Thematic fame, melodic originality, and musical zeitgeist: A biographical and transhistorical content analysis. *Journal of Personality and Social Psychology*, 38(6):972–983, 1980. doi: 10.1037/0022-3514.38.6.972.

[182] Dean Keith Simonton. Emergence and realization of genius: The lives and works of 120 classical composers. *Journal of Personality and Social Psychology*, 61(5):829–840, 1991. doi: 10.1037/0022-3514.61.5.829.

[183] Malcolm Slaney and Michael A. Casey. Locality-sensitive hashing for finding nearest neighbors. *Signal Processing Magazine, IEEE*, 25(2):128–131, 2008. doi: 10.1109/MSP.2007.914237.

[184] Marko Stamenovic. Towards cover song detection with siamese convolutional neural networks. *CoRR*, abs/2005.10294, 2020.

[185] Daniel Stoller, Simon Durand, and Sebastian Ewert. End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 181–185, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8683470.

[186] Li Su. Vocal melody extraction using patch-based CNN. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 371–375, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8462420.

[187] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, Montréal, Canada, 2014.

[188] Iman S.H. Suyoto, Alexandra L. Uitdenbogerd, and Falk Scholer. Searching musical audio using symbolic queries. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):372–381, 2008. doi: 10.1109/TASL.2007.911644.

[189] Aaron Swartz. MusicBrainz: A semantic web service. *IEEE Intelligent Systems*, 17(1):76–77, 2002. doi: 10.1109/5254.988466.

[190] John Thickstun, Zaïd Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[191] Renee Timmers, Nicola Dibben, Zohar Eitan, Roni Granot, Tim Metcalfe, Andrea Schiavio, and Victoria Williamson. Introduction to the proceedings of ICMEM 2015. In *Proceedings of the International Conference on the Multimodal Experience of Music (ICMEM)*, Sheffield, UK, 2015.

[192] Christopher J. Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 294–301, Suzhou, China, 2017. doi: 10.5281/zenodo.1417331.

[193] Timothy Tsai, Thomas Prätzlich, and Meinard Müller. Known-artist live song identification using audio hashprints. *IEEE Transactions on Multimedia*, 19(7):1569–1582, 2017. doi: 10.1109/TMM.2017.2669864.

[194] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 153–160, London, UK, 2005. doi: 10.5281/zenodo.1417383.

[195] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002. doi: 10.1109/TSA.2002.800560.

[196] Alexandra L. Uitdenbogerd and Yaw Wah Yap. Was Parsons right? An experiment in usability of music representations for melody-based music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Baltimore, Maryland, USA, 2003. doi: 10.5281/zenodo.1418225.

[197] Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gaël Richard, and Florence d'Alché-Buc. Multilingual lyrics-to-audio alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 512–519, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245484.

[198] Harsh Verma and John Thickstun. Convolutional composer classification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 549–556, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527866.

[199] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[200] Anja Volk, Frans Wiering, and Peter Van Kranenburg. Unfolding the potential of computational musicology. In *Proceedings of the International Conference on Informatics and Semiotics in Organisations (ICISO)*, pages 137–144, Leeuwarden, The Netherlands, 2011.

[201] Piet G. Vos and Jim M. Troost. Ascending and descending melodic intervals: Statistical findings and their perceptual relevance. *Music Perception*, 6(4):383–396, 1989. doi: 10.2307/40285439.

[202] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 7–13, Baltimore, Maryland, USA, 2003. doi: 10.5281/zenodo.1416340.

[203] Avery Wang. The Shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006. doi: 10.1145/1145287.1145312.

[204] Christof Weiß, Frank Zalkow, Meinard Müller, Stephanie Klauk, and Rainer Kleinertz. Versionsübergreifende Visualisierung harmonischer Verläufe: Eine Fallstudie zu Wagners Ring-Zyklus. In *Proceedings of the Jahrestagung der Gesellschaft für Informatik (GI)*, pages 205–217, Chemnitz, Germany, 2017. doi: 10.18420/in2017_14.

[205] Christof Weiß, Frank Zalkow, Vlora Arifi-Müller, Meinard Müller, Hendrik Vincent Koops, Anja Volk, and Harald G. Grohganz. Schubert Winterreise dataset: A multimodal scenario for music analysis. *ACM Journal on Computing and Cultural Heritage (JOCCH)*, 14(2), 2021. doi: 10.1145/3429743.

[206] Yiming Wu and Wei Li. Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 27(2): 355–366, 2019. doi: 10.1109/TASLP.2018.2879399.

[207] Yiming Wu, Tristan Carsault, and Kazuyoshi Yoshii. Automatic chord estimation based on a frame-wise convolutional recurrent neural network with non-aligned annotations. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 1–5, A Coruña, Spain, 2019. doi: 10.23919/EUSIPCO.2019.8902741.

[208] Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Key-invariant convolutional neural network toward efficient cover song identification. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, San Diego, California, USA, 2018. doi: 10.1109/ICME.2018.8486531.

[209] Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Effective cover song identification based on skipping bigrams. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 96–100, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8462404.

[210] Furkan Yesiler, Joan Serrà, and Emilia Gómez. Less is more: Faster and better music version identification with embedding distillation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 884–892, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245570.

[211] Furkan Yesiler, Joan Serrà, and Emilia Gómez. Accurate and scalable version identification using musically-motivated embeddings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 21–25, Barcelona, Spain, 2020. doi: 10.1109/ICASSP40776.2020.9053793.

[212] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Temporal pyramid pooling convolutional neural network for cover song identification. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 4846–4852, Macao, China, 2019. doi: 10.24963/ijcai.2019/673.

[213] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang. Learning a representation for cover song identification using convolutional neural network. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 541–545, Barcelona, Spain, 2020. doi: 10.1109/ICASSP40776.2020.9053839.

[214] Frank Zalkow and Meinard Müller. Vergleich von PCA- und Autoencoder-basierter Dimensionsreduktion von Merkmalssequenzen für die effiziente Musiksuche. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1526–1529, Munich, Germany, 2018.

[215] Frank Zalkow and Meinard Müller. Using weakly aligned score–audio pairs to train deep chroma models for cross-modal music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 184–191, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245400.

[216] Frank Zalkow and Meinard Müller. Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music. *Applied Sciences*, 10(1), 2020. doi: 10.3390/app10010019.

[217] Frank Zalkow and Meinard Müller. CTC-based learning of deep chroma features for score–audio music retrieval. 2021. Currently under review.

[218] Frank Zalkow, Stephan Brand, and Bejamin Graf. Musical style modification as an optimization problem. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 206–211, Utrecht, The Netherlands, 2016.

[219] Frank Zalkow, Christof Weiß, and Meinard Müller. Exploring tonal-dramatic relationships in Richard Wagner's Ring cycle. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 642–648, Suzhou, China, 2017. doi: 10.5281/zenodo.1415760.

[220] Frank Zalkow, Christof Weiß, Thomas Prätzlich, Vlora Arifi-Müller, and Meinard Müller. A multi-version approach for transferring measure annotations between music recordings. In *Proceedings of the AES International Conference on Semantic Audio*, pages 148–155, Erlangen, Germany, 2017.

[221] Frank Zalkow, Stefan Balke, and Meinard Müller. Evaluating salience representations for cross-modal retrieval of Western classical music recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 331–335, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8683609.

[222] Frank Zalkow, Stefan Balke, Vlora Arifi-Müller, and Meinard Müller. MTD: A multimodal dataset of musical themes for MIR research. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1):180–192, 2020. doi: 10.5334/tismir.68.

[223] Frank Zalkow, Julian Brandner, and Meinard Müller. Efficient retrieval of music recordings using graph-based index structures. *Signals*, 2(2):336–352, 2021. doi: 10.3390/signals2020021.

[224] Wei Zhang, Ting Yao, Shiai Zhu, and Abdulmotaleb El-Saddik. Deep learning-based multimedia analytics: A review. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15 (1s):2:1–2:26, 2019. doi: 10.1145/3279952.

[225] Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4): 265–278, 2016. doi: 10.1016/j.jfds.2017.05.001.