

Dissertation

2026

Johannes Christian Zeitler

Differentiable Alignment Techniques for Music Processing



Differentiable Alignment Techniques for Music Processing

Differenzierbare Alignierungstechniken für die Musikverarbeitung

Dissertation

Der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

zur Erlangung des Doktorgrades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von
Johannes Christian Zeitler

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

Tag der mündlichen Prüfung: 25.09.2026

Gutachter: Prof. Dr. rer. nat. Meinard Müller
Prof. Dr. Geoffroy Peeters

Statement on AI-Assisted Editing

The author used ChatGPT (<https://chatgpt.com>), a large language model, to improve spelling, grammar, clarity, and readability during the preparation of this dissertation. The tool was applied exclusively for language editing and was not used to generate, outline, or summarize any portion of the content. The author carefully reviewed and edited the final version of the text and takes full responsibility for the content of this dissertation.

Abstract

Music Information Retrieval (MIR) seeks to develop computational methods for analyzing, organizing, and interacting with music across heterogeneous representations such as audio recordings, symbolic scores, and musicological annotations. A central technical challenge underlying many MIR tasks is temporal alignment: different representations of the same musical work are encoded on distinct time axes and must be synchronized to enable joint processing. From a computer science perspective, this problem can be formulated as sequence alignment under domain-specific constraints. At the same time, modern data-driven approaches rely on neural networks trained with gradient-based optimization and typically require strong frame-level annotations, which are costly to obtain and often unavailable at large scales.

This thesis studies differentiable alignment techniques as a framework that connects classical alignment methods with modern deep learning in MIR. Building on Dynamic Time Warping (DTW), a well-established alignment algorithm for time series, we show how intermediate representations from neural transcription models can serve as robust alignment features, improving the accuracy and reliability of DTW-based synchronization in challenging cross-version and cross-modal settings. We demonstrate the practical impact of this approach in the curation of the Beethoven Piano Sonata Dataset (BPSD), where scores, multiple performances, and musicological annotations are consolidated on a shared temporal axis to support the training and testing of machine learning approaches and computational musicology.

Beyond using alignment in post-processing, as its main technical contribution, this thesis advances sequence alignment as a component in end-to-end differentiable learning pipelines. We analyze and extend Soft Dynamic Time Warping (SDTW), a differentiable relaxation of DTW that enables training from weakly aligned data, and improve its practical usability through stabilization strategies and extensions such as variable step weights and relaxed boundary conditions, including subsequence alignment. To better understand and control its behavior, we reformulate the SDTW gradient and interpret differentiable alignment as cost aggregation over paths in weighted graphs. This graph-based view clarifies validity conditions for Dynamic Programming (DP) recursions, relates SDTW to Connectionist Temporal Classification (CTC), and leads to Differentiable Dynamic Time Warping (dDTW), a unified framework that recovers DTW, SDTW, and CTC as special cases.

By linking alignment, graph-based optimization, and gradient-based learning, this thesis advances differentiable sequence alignment as both an algorithmic tool and a training objective. The results contribute to computer science through a unified and extensible view of differentiable alignment, and to MIR through robust synchronization methods and effective learning from weak temporal supervision.

Zusammenfassung

Das Forschungsgebiet Music Information Retrieval (MIR) entwickelt rechnergestützte Methoden zur Analyse und Organisation von Musik über unterschiedliche Darstellungen hinweg, wie etwa Audioaufnahmen, symbolische Notentexte und musikwissenschaftliche Annotationen. Eine zentrale Herausforderung ist die zeitliche Alignierung: Verschiedene Darstellungen desselben Werkes sind auf unterschiedlichen Zeitachsen kodiert und müssen zur gemeinsamen Verarbeitung synchronisiert werden. Aus Sicht der Informatik und Signalverarbeitung entspricht dies einer Sequenz-Synchronisation unter domänenspezifischen Nebenbedingungen. Dies ist besonders relevant für das Training moderner neuronaler Verfahren, welches meist zeitlich exakte Annotationen erfordert. Eine manuelle Erstellung dieser Annotationen ist jedoch aufwendig und häufig nicht skalierbar.

Diese Dissertation untersucht differenzierbare Alignierungstechniken, mittels derer klassische Synchronisationsmethoden in maschinelle Lernverfahren integriert werden können. Ausgehend von Dynamic Time Warping (DTW) zeigen wir, wie Zwischenrepräsentationen neuronaler Transkriptionsmodelle als robuste Merkmale die Genauigkeit DTW-basierter Synchronisation über verschiedene Versionen und Modalitäten hinweg verbessern. Die praktische Relevanz dieses Synchronisationsverfahrens wird bei der Erstellung des Beethoven Piano Sonata Dataset (BPSD) demonstriert, in dem Notentext, Audioaufnahmen und Annotationen mittels einer gemeinsamen Zeitachse in Verbindung gesetzt werden können.

Zentraler Beitrag dieser Arbeit ist die Erforschung der Sequenzalignierung als Bestandteil differenzierbaren Lernens. Insbesondere analysieren und erweitern wir Soft Dynamic Time Warping (SDTW), eine differenzierbare Variante von DTW, die End-to-End-Training mit schwach alignierten Daten ermöglicht, und verbessern die Stabilität durch variable Schrittweitengewichte und flexible Randbedingungen einschließlich Subsequenz-Alignierung. Eine graphenbasierte Interpretation präzisiert die Gültigkeitskriterien der dynamischen Programmierung, stellt den Bezug zu Connectionist Temporal Classification (CTC) her und führt zu Differentiable Dynamic Time Warping (ddTW), einer vereinheitlichten Darstellung, die DTW, SDTW und CTC als Spezialfälle umfasst.

Diese Dissertation verbindet Alignierung, graphenbasierte Optimierung sowie gradientenbasiertes Lernen und entwickelt Sequenzalignierung sowohl als algorithmisches Werkzeug als auch als Kostenfunktion weiter. Sie leistet damit einen Beitrag in den Bereichen der Informatik und Signalverarbeitung und stellt durch robuste Synchronisationstechniken sowie maschinelles Lernen mit schwacher zeitlicher Supervision Verfahren zur MIR bereit.

Contents

Abstract	vii
Zusammenfassung	ix
1 Introduction	1
1.1 Structure and Main Contributions	3
1.2 Publications Underlying this Thesis	5
1.3 Additional Contributions	6
1.4 Acknowledgments	7
2 Background: Music Synchronization	9
2.1 Overview	9
2.2 Feature Extraction and Representation	11
2.3 Cost Measures	13
2.4 Dynamic Time Warping (DTW)	14
2.5 Robust, Efficient, and High-Accuracy Synchronization	19
2.6 Experiment: Music Synchronization with Neural Features	20
3 Beethoven Piano Sonata Dataset: Alignment-Based Dataset Curation	27
3.1 Introduction	28
3.2 Related Datasets and Resources	30
3.3 Data Organization and Primary Musical Material	33
3.4 Music Synchronization	37
3.5 Annotations	41
3.6 Application Scenarios	44
3.7 Further Notes	46
4 Background: Soft Dynamic Time Warping (SDTW)	49
4.1 Related Work: Deep Learning and Differentiable Alignment	49

Contents

4.2	Differentiable Approximation of the Minimum Function: Softmin	50
4.3	SDTW: Global Formulation	52
4.4	SDTW: Local Formulation	55
4.5	Limit Cases	59
5	Stabilizing SDTW for End-to-End Training	63
5.1	Motivation and Related Work	63
5.2	Stabilizing Training with SDTW	65
5.3	Experimental Setup	66
5.4	Evaluation	68
5.5	Further Notes	72
6	SDTW with Variable Step Weights	73
6.1	Motivation and Related Work	73
6.2	Weighted SDTW	74
6.3	Experiments and Evaluation	79
6.4	Further Notes	82
7	Subsequence Variants of SDTW	85
7.1	Motivation for Subsequence Alignment	85
7.2	Subsequence SDTW	86
7.3	Experiments	91
7.4	Further Notes	94
8	Understanding SDTW: Target Artifacts and Prediction Quality	95
8.1	Problem Statement and Observed Artifacts	95
8.2	Gradient Reformulation into Modified Targets	97
8.3	Evaluation	101
8.4	Further Notes	104
9	Unifying Views: CTC and Differentiable DTW	105
9.1	Motivation and Related Work	106
9.2	CTC Revisited	107
9.3	SDTW Revisited	109
9.4	Relating CTC and SDTW	111
9.5	Differentiable Dynamic Time Warping	113
9.6	Label Probabilities and Cost Functions	119
9.7	Experiments	123
9.8	Further Notes	134

10 A Graph Perspective on Differentiable Alignment	137
10.1 Graphs and Shortest-Path Problems	137
10.2 Sequence Alignment as a Shortest-Path Problem on Graphs	138
10.3 Solving the Minimum-Cost Problem with Dynamic Programming	141
10.4 Practical Computation of dDTW Cost and Gradient	142
10.5 Examples for Alignment Problems Generalized by dDTW	147
10.6 Further Notes	151
11 Summary and Future Work	153
 Appendix	 155
A Convex Optimization	155
B Differentiable Minimum Functions	161
C dDTW Toolbox	179
 List of Figures	 183
 List of Tables	 186
 List of Acronyms	 187
 List of Symbols	 188
 Bibliography	 191

1 Introduction

Music Information Retrieval (MIR) lies at the intersection of engineering, computer science, the arts, and digital humanities. It develops computational methods for analyzing, organizing, and interacting with music. From a humanities perspective, computational tools support musicological research and artistic practice. From an engineering perspective, music provides a rich and challenging domain for developing novel signal processing and machine learning techniques.

A central feature of music data is that the same underlying semantic content often exists in multiple representations. A composed work may appear as a score, as edited or adapted versions, and as performances by different musicians, each characterized by individual choices of tempo, dynamics, articulation, expressive timing, and sometimes instrumentation. Performances are typically available as audio recordings, but may also be represented symbolically, for example as Musical Instrument Digital Interface (MIDI) files encoding note events and control changes. In addition, music collections often contain annotations such as harmony labels, structural markers, or lyrics. Many MIR tasks require the joint processing of multiple representations of the same musical work. Performance analysis and score following rely on modeling correspondences between audio recordings and symbolic scores [3, 31, 33, 37, 144, 161]. Lyrics following and karaoke systems align recordings with textual annotations [135]. Interactive applications synchronize and display recordings with structural and harmonic annotations [84, 161]. While approaches for self-supervised learning exist in MIR [43, 51, 122, 138], precise audio–symbolic correspondences are also required to train supervised systems for Automatic Music Transcription (AMT) and other learning-based feature extraction tasks [12, 148]. More generally, cross-version or cross-modal analysis investigates differences across performances or representations of the same work [38, 150].

A unifying perspective from an engineering and computer science viewpoint is that these data sources can be modeled as time-dependent sequences. Audio recordings form continuous time series; scores define sequences of musical events structured in beats and measures; symbolic formats use quantized timing such as MIDI ticks; annotations evolve along their own temporal axes, depending on the type of annotation. Each representation defines its own notion of time, while representing the same semantic content. Consequently, different versions of the same piece follow related but not identical timelines. In this thesis, we investigate the *alignment*, or *synchronization*, between these sequences, using the two terms synonymously throughout. We distinguish between *strongly aligned* and *weakly aligned* sequences. Sequences are strongly aligned if their events correspond exactly on a shared timeline. In contrast, sequences are weakly aligned if they describe the same musical content at a segment level, sharing a

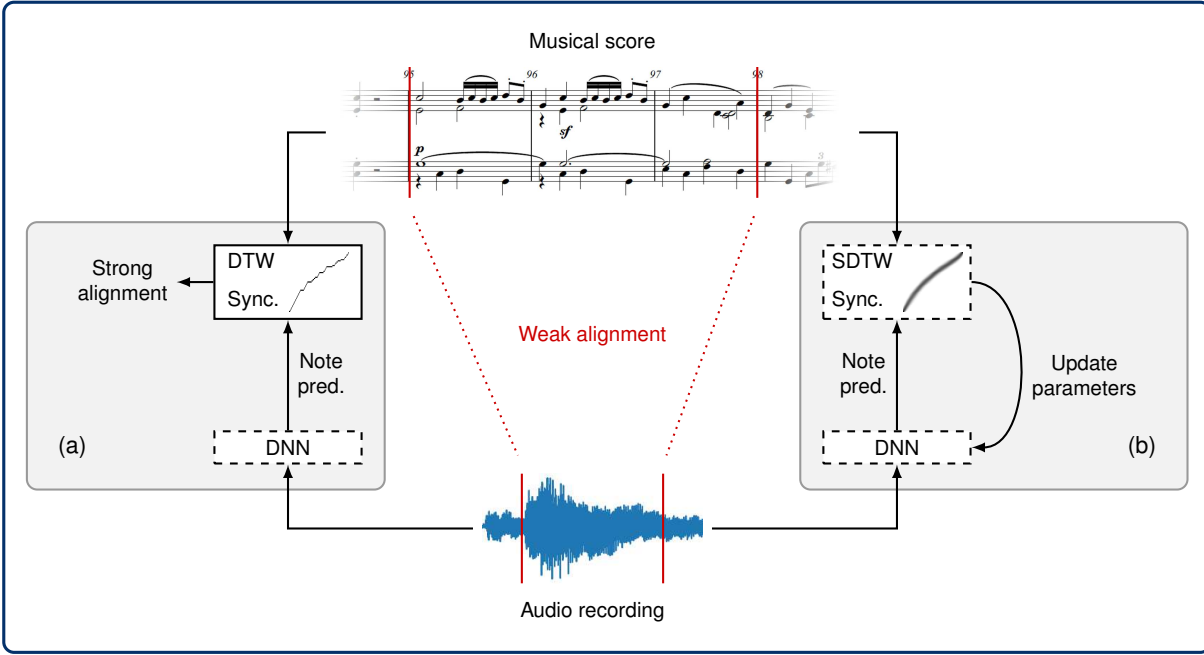


Figure 1.1: Schematic overview of synchronizing weakly aligned score–audio pairs (center) using differentiable techniques (indicated by dashed lines). (a) Features extracted by a differentiable model serve as input to a classical Dynamic Time Warping (DTW)-based alignment method, yielding a strong alignment. (b) The Soft Dynamic Time Warping (SDTW) loss provides a fully differentiable alignment objective for training a DNN from weakly aligned data.

start point, an end point, and the order of events, but differ in their local timing or overall temporal structure. Given weakly aligned sequences, music synchronization (or alignment) aims to compute a temporal mapping between their timelines (see Figure 1.1). While manual annotation of corresponding time points is possible, it is impractical at scale [149]. Automated alignment methods therefore play a central role in creating and understanding semantic relationships in heterogeneous music collections. A classical solution is Dynamic Time Warping (DTW), which computes a cost-minimizing alignment under monotonic warping constraints using efficient Dynamic Programming (DP) recursions [98].

At the same time, differentiable and data-driven methods have become dominant in MIR. Deep Neural Networks (DNNs) can be trained for tasks such as transcription [50, 63, 65, 77], chord recognition [80, 81, 157], melody estimation [6, 10, 14, 73], and lyrics alignment [60, 135, 142]. These systems are optimized end-to-end using gradient descent strategies [52]. This development creates two complementary opportunities for music synchronization. First, learned representations can serve as robust features for classical alignment algorithms, enabling high-quality strong alignments that support downstream MIR tasks (see Figure 1.1a). Second, alignment itself can be embedded as a differentiable component within machine learning pipelines, enabling training from weakly aligned data (see Figure 1.1b). This thesis investigates both perspectives.

The first contribution of this thesis integrates feature representations derived from trained neural models into classical synchronization pipelines. We analyze how intermediate representations from Deep

Learning (DL)-based transcription systems improve the robustness and accuracy of DTW-based alignment for piano recordings with challenging acoustic conditions. We further demonstrate that fine-tuning transcription models on target data enhances robustness without overfitting. Building on these results, we apply the proposed pipeline to curate the Beethoven Piano Sonata Dataset (BPSD) by aligning scores, performances, and annotations on a shared timeline, thereby enabling consistent cross-modal analysis and training.

The second contribution, which is the central focus of this thesis, addresses end-to-end learning from weakly aligned input–target pairs. In many training scenarios, input and target correspond at a global level but differ in local timing, preventing frame-wise supervision. Existing strategies for learning from weakly aligned data include generating strong alignments through semi-automatic pipelines [103, 153, 155] or iteratively refining alignments and model predictions in Expectation–Maximization (EM)-like procedures [90, 121, 141, 159]. An alternative is to incorporate a differentiable alignment mechanism directly into the loss function. Alignment paradigms used for this purpose include attention mechanisms [143], recurrent neural networks [57], optimal transport formulations [29, 139], and Connectionist Temporal Classification (CTC) [59]. In this thesis, we focus on differentiable extensions of DTW, in particular SDTW [30], which provides a conceptually simple alignment loss applicable to arbitrary feature representations. While SDTW enables learning under temporal misalignment, its optimization behavior can be unstable and sensitive to design choices [82].

As the main technical contribution of this thesis, we analyze and advance SDTW for efficient and robust training of DNNs in weakly supervised scenarios. We propose stabilization strategies for early training, generalize SDTW to incorporate classical DTW extensions such as variable step weights and relaxed boundary conditions, and provide an interpretable analysis of its internal alignment structure. Furthermore, we establish a theoretical connection between SDTW and CTC and propose a unified differentiable alignment framework that generalizes both methods. Finally, we present a graph-based perspective on alignment, yielding validity guarantees for the resulting DP recursions and a structured view on differentiable relaxations of related algorithms.

1.1 Structure and Main Contributions

This thesis is organized as follows. In Chapter 2, we review classical alignment approaches in MIR, with a focus on feature representations, DTW, and its extensions. In Section 2.6, we combine classical DTW-based alignment with features extracted from state-of-the-art DL-based transcription models. We show that intermediate predictions for frame and onset probabilities improve synchronization accuracy and robustness when integrated into existing high-resolution alignment pipelines. In Chapter 3, we apply this alignment pipeline to curate a large-scale classical music dataset, the Beethoven Piano Sonata Dataset (BPSD). The BPSD contains sheet music and musicological annotations for the first movements of all 32 piano sonatas by Ludwig van Beethoven, together with audio recordings of eleven performances.

Chapter 1. Introduction

Using the synchronization techniques from Section 2.6, we align scores, recordings, and annotations on a shared musical timeline. We evaluate the resulting alignments in a transcription scenario, a harmony analysis task, and cross-version chord recognition strategies.

The main technical contributions of this thesis concern extensions and analyses of SDTW as a loss function for end-to-end training of DL models. In Chapter 4, we provide the background and lay the foundation for fully differentiable synchronization. To this end, we introduce SDTW as a differentiable relaxation of DTW and derive the SDTW cost and gradient computations in both global and DP formulations. In Chapter 5, we identify a mismatch between early soft alignments and reference alignments as a key source of training instabilities. We propose three complementary strategies to stabilize intermediate soft alignments. In Chapter 6, we extend SDTW to support variable step weights. This modification improves stability in settings with strongly differing or exactly identical sequence lengths. We derive efficient forward and backward DP recursions for weighted SDTW and demonstrate its effectiveness when training a DNN for Pitch Class Estimation (PCE). In Chapter 7, we relax the boundary conditions of SDTW and introduce a subsequence variant, termed subSDTW. This extension supports training from input–target pairs with less precise temporal correspondence. In Chapter 8, we analyze the internal mechanisms of the SDTW loss. We provide a reformulation that represents the SDTW gradient as an equivalent strongly aligned target representation. This view enables an interpretable analysis of learned representations, highlights the effect of SDTW hyperparameters, and explains artifacts commonly observed during training.

In Chapter 9, we move beyond extensions of SDTW toward a more general view of differentiable alignment. We establish a theoretical connection between SDTW and CTC, a standard loss for training Automatic Speech Recognition (ASR) systems from weakly aligned data. Building on this connection, we introduce a generalized differentiable alignment loss, termed Differentiable Dynamic Time Warping (dDTW), which supports cell-wise step weights, flexible step sizes, boundary conditions, and minimum operators. We show that dDTW subsumes SDTW and CTC as special cases and compare these losses in two controlled MIR tasks. In Chapter 10, we further abstract this perspective by formulating alignment as a shortest-path problem on graphs. This view unifies the underlying DP recursions of SDTW and its variants, clarifies their relationship to CTC, and provides a general framework in which differentiable relaxations of other alignment paradigms, such as Partial Matching (PM), can be expressed.

We conclude in Chapter 11 with a summary of the main results and a discussion of directions for future work. In Appendices A to C, we provide further background on convex optimization, differentiable minimum functions, and a practical implementation of the dDTW loss.

1.2 Publications Underlying this Thesis

The following peer-reviewed publications form the scientific foundation of this thesis and document its main research contributions. Johannes Zeitler is the main author and contributor to all these publications.

- [169] Johannes Zeitler, Ben Maman, and Meinard Müller. Robust and accurate audio synchronization using raw features from transcription models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 120–127, San Francisco, USA, 2024. doi: 10.5281/ZENODO.14877291

This article is part of Chapters 2 and 3 and was also printed in [89]. Johannes Zeitler is the main contributor to this article, responsible for code implementation, evaluation, experiments, analysis, and writing. Ben Maman trained models used for evaluation. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

- [170] Johannes Zeitler, Christof Weiß, Vlora Arifi-Müller, and Meinard Müller. BPSD: A coherent multi-version dataset for analyzing the first movements of beethoven’s piano sonatas. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 7(1):195–212, 2024. doi: 10.5334/TISMIR.196

This article forms the basis of Chapter 3. Compiling the BPSD involved an extensive workflow, encompassing the preparation and processing of raw data, typesetting, manual corrections, annotation creation and transfer, as well as the utilization of the data for research. This complex process underwent multiple cycles of revision and correction, requiring a collaborative effort from a considerable number of individuals within the research team. All co-authors of the paper [170] actively participated in both the data curation process and the preparation of the final manuscript. On the technical level, Johannes Zeitler is the main contributor to the design, implementation, and evaluation of the alignment strategy, the consolidation of the annotations, and the implementation of the application-oriented case studies.

- [167] Johannes Zeitler, Simon Deniffel, Michael Krause, and Meinard Müller. Stabilizing training with soft dynamic time warping: A case study for pitch class estimation with weakly aligned targets. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–439, Milano, Italy, 2023. doi: 10.5281/ZENODO.10265317

This article forms the basis of Chapter 5. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Simon Deniffel validated the approaches as part of his research project. Michael Krause shared his expertise in training with the SDTW loss function. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

- [168] Johannes Zeitler, Michael Krause, and Meinard Müller. Soft dynamic time warping with variable step weights. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 356–360, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10446578

This article forms the basis of Chapter 6. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Michael Krause shared his expertise in training with the SDTW loss function. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

- [166] Johannes Zeitler and Meinard Müller. Subsequence SDTW: Differentiable alignment with flexible boundary conditions. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 15252–15256, Barcelona, Spain, 2026. doi: 10.1109/ICASSP55912.2026.11463857

This article forms the basis of Chapter 7. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

Chapter 1. Introduction

- [164] Johannes Zeitler and Meinard Müller. Reformulating soft dynamic time warping: Insights into target artifacts and prediction quality. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–133, Daejeon, South Korea, 2025. doi: 10.5281/ZENODO.17706349

This article forms the basis of Chapter 8. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

- [165] Johannes Zeitler and Meinard Müller. A unified perspective on CTC and SDTW using differentiable DTW. *IEEE Transactions on Audio, Speech and Language Processing*, 34:936–951, 2026. doi: 10.1109/TASLPRO.2026.3657213

This article forms the basis of Chapter 9. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

1.3 Additional Contributions

In addition to the main articles that form the core of this thesis, the author of this thesis has also contributed to the following accepted or submitted publications in the field of Music Information Retrieval.

- [154] Christof Weiß, Johannes Zeitler, Tim Zunner, Florian Schuberth, and Meinard Müller. Learning pitch-class representations from score–audio pairs of classical music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 746–753, Online, 2021. doi: 10.5281/zenodo.5624549
- [91] Ben Maman, Johannes Zeitler, Meinard Müller, and Amit H. Bermanno. Performance conditioning for diffusion-based multi-instrument music synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5045–5049, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10445979
- [27] Ching-Yu Chiu, Johannes Zeitler, Vlora Arifi-Müller, and Meinard Müller. Downbeat tracking for Western classical music recordings: A case study for Beethoven piano sonatas. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1–4, Hannover, Germany, 2024
- [92] Ben Maman, Johannes Zeitler, Meinard Müller, and Amit H. Bermanno. Multi-aspect conditioning for diffusion-based music synthesis: Enhancing realism and acoustic control. *IEEE Transactions on Audio, Speech, and Language Processing*, 33:68–81, 2025. doi: 10.1109/TASLP.2024.3507553
- [36] Christian Dittmar, Johannes Zeitler, Stefan Balke, Simon Schwär, and Meinard Müller. PULSE-IT: Lightweight and expressive synthesis of wind instrument playing. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Daejeon, South Korea, 2025
- [104] Meinard Müller, Johannes Zeitler, and Christof Weiß. A computational approach for visualizing tonal structures in music recordings. In *Oxford Handbook for Systematic Empirical Research in the Arts*, Oxford Handbooks. Oxford University Press, accepted 2025
- [105] Meinard Müller, Johannes Zeitler, and Sebastian Strahl. PCPT notebooks: A preparation course for PyTorch with a focus on signal processing. *Journal of Open Source Education (JOSE)*, submitted 2025
- [7] Stefan Balke, Johannes Zeitler, Vlora Arifi-Müller, Brian McFee, Tomoyasu Nakano, Masataka Goto, and Meinard Müller. RWC revisited: Towards a community-driven MIR corpus. *Transactions of the International Society for Music Information Retrieval*, 9:21–35, 2026. doi: 10.5334/tismir.326

1.4 Acknowledgments

I am deeply grateful for the opportunity to pursue my PhD at the International Audio Laboratories Erlangen, a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer Institute for Integrated Circuits IIS. I would also like to thank the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for funding my PhD under Grants No. 252013209 (MU 2686/7-2, CAS), 521420645 (MU 2686/17-1, DAT4MIR), and 500643750 (MU 2686/15-1, LEARN). Having such an outstanding environment and secure funding is a true privilege and allowed me to fully focus on my research.

Most of all, I would like to thank my supervisor Meinard for his exceptional support. Thank you, Meinard, for responding to my “emergency call” and offering me a position in your group—I truly could not have made a better choice. Your dedication to supporting every member of the group is remarkable: you provide clear guidance and direction, engineer connections and synergies, invest tremendous effort into diving into and carefully reviewing our work (even if the “Meinard revisions” can be challenging, they are always worth it), handle administrative matters so we can focus on research, and create an environment that supports us far beyond academia. Many, many thanks!

I would also like to express my sincere thanks to the team working behind the scenes, who ensure that everything runs smoothly, from daily operations to long-term strategy. In alphabetical order, I would like to thank: Carmen Craciun, Christian Dittmar, Bernd Edler, Bernhard Grill, Emmanuël Habets, Tracy Harris, Jürgen Herre, Meinard Müller, Frederik Nagel, Petra Neubarth, Nils Peters, Day-See Riechmann, Stefan Turowski, and Elke Weiland.

The AudioLabs would not be the same without its wonderful colleagues, whom I would like to thank here: Alexander Adami, Ahmad Aloradi, Carlotta Anemüller, Pablo Delgado, Sascha Dick, Mohamed Elminshawy, Richard Füg, Ünal Ege Gaznepoğlu, Philipp Götz, Ning Guo, Alexander Heinmüller, Adrian Herzog, Yaqiong Hou, Weilong Huang, Guanxin Jiang, Thorsten Kastner, Srikanth Korse, Kishor Kayyar Lakshminarayana, Jeremy Lawrence, Anna Leschanowsky, Wolfgang Mack, Daniele Mirabilii, Thomas Robotham, Dipanjan Datta Roy, Lorenz Schmidt, Martin Strauss, Matteo Torcoli, Anika Treffehn, Julian Wechsler, Nils Werner, Niklas Winter, and Zeyu Xu. I would especially like to thank my colleagues from “GroupMM”: Jakob Abeßer, Vlora Arifi-Müller, Stefan Balke, Judith Bauer, Hans-Ulrich Berendes, Ching-Yu “Sunny” Chiu, Christian Dittmar, Jueun Kang, Michael Krause, Ben Maman, Peter Meier, Yiğitcan Özer, Manuel Peters, Sebastian Rosenzweig, Abhirup Saha, Simon Schwär, Sebastian Strahl, Christof Weiß, Philipp Weyers, Frank Zalkow. It has been truly special to be part of such a wonderful group—not only colleagues, but close friends. Thank you for all the moments we shared: preparing premium coffee, attending conferences, going on ski trips and hikes, researching “beach enhancement”, celebrating together, enjoying concerts, making Kaiserschmarrn, and expanding our “athletic discovery.”

I would also like to thank the colleagues and friends from my former work group, from whom I learned a lot: Andreas Brendel, Annika Briegleb, Alexander Briegleb-Schmidt, Cornelius Frankenbach, Michael

Chapter 1. Introduction

Günther, Modar Halimeh, Thomas Haubner, Matthias Kreuzer, and Heiner Löllmann. Thank you for encouraging me to pursue my PhD at the AudioLabs—it was indeed the perfect choice.

I am grateful to all fellow researchers I had the opportunity to meet and learn from during my PhD. In particular, I would like to thank Prof. Geoffroy Peeters for many insightful discussions and for reviewing this thesis. I would also like to thank the ISMIR community for providing an inspiring platform for MIR research, with the yearly ISMIR conferences being a particular highlight. A special thanks goes to the organizers of the wonderful ISMIR 2025 conference in Daejeon, South Korea, Profs. Juhan Nam and Dasaem Jeong—it was an outstanding event. Many thanks also to their students, whose dedication made this experience possible. I would especially like to thank Jaeran Choi, Taegyun Kwon, Sein Lee, and Jyiun Park. I am very grateful to have met you multiple times during my PhD and to call you friends. Thank you for sharing insights into your lives, culture, and research, for helping us feel at home in a foreign country, and for discovering fantastic restaurants and bars together.

I thank Hans-Ulrich Berendes, Ben Maman, Manuel Peters, Abhirup Saha, Simon Schwär, Sebastian Strahl, and of course Meinard Müller for proof-reading parts of this thesis.

Finally, I would like to thank my wonderful family for their love and support. Without my grandparents, parents, my sister, and my wife, I would not be who I am today. Your constant encouragement, unwavering support, and inspiring example gave me the freedom and confidence to grow, learn, and follow my own path without worry. Thank you!

Erlangen, March 2026

2 Background: Music Synchronization

This chapter is partly based on [169]. Johannes Zeitler is the main contributor to this article, responsible for code implementation, evaluation, experiments, analysis, and writing. Ben Maman trained models used for evaluation. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

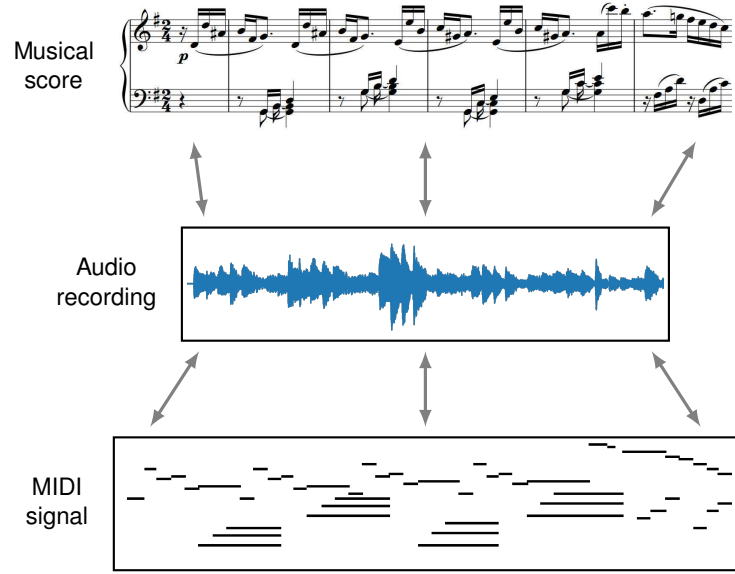
Music synchronization establishes temporal correspondences between different versions of the same musical piece, for example between symbolic score representations, audio recordings, MIDI files, and annotations, such that musical events align in time across versions or modalities (see Figure 2.1). In Section 2.1, we introduce the task of music synchronization and give an overview of typical synchronization pipelines, followed by a discussion of feature extraction and representation in Section 2.2. In Section 2.3, we review common pairwise cost measures used in music synchronization and machine learning. We then introduce DTW in Section 2.4, which serves as the conceptual foundation for sequence alignment throughout this thesis, and give an overview of DTW extensions in Section 2.5. Finally, Section 2.6 presents a first synchronization experiment highlighting the importance of suitable feature extractors and alignment strategies, and proposes a novel approach based on raw output features from neural transcription models.

2.1 Overview

Aligning different versions or representations of the same musical work is a fundamental task in MIR. In score–audio synchronization, symbolic note information from a score is aligned with the timeline of an audio recording. This underlies applications such as automatic score following [3, 130], score-informed source separation [46], and the generation of note-level annotations for training and evaluating AMT systems [90]. Similarly, audio–audio synchronization aligns multiple recordings of the same piece. This enables applications including track switching in interactive systems [161], cross-version analysis [35], automatic accompaniment based on pre-recorded material [33, 114], and the transfer of annotations between performances [153, 155].

Chapter 2. Background: Music Synchronization

Figure 2.1: Illustration of a music synchronization scenario with different modalities. The musical score, the audio recording, and a MIDI signal represent the same semantic information, but they are represented on different timelines.



Abstractly, music synchronization seeks to align two time-dependent sequences while preserving their temporal order. Local time stretching and compression are permitted to account for expressive timing differences, whereas reordering or skipping of events is typically disallowed in sequence alignment.

Classical alignment pipelines generally follow a two-stage procedure (see Figure 2.2). First, the weakly aligned input data are transformed into suitable feature representations that allow for meaningful comparison. For instance, audio signals may be converted into time–frequency representations such as Short-Time Fourier Transform (STFT) or Constant-Q Transform (CQT) features, whereas symbolic scores may be discretized into sequences of note events (see Section 2.2). Second, an alignment algorithm synchronizes the feature sequences, producing a strong alignment between the original data representations (see Sections 2.3 and 2.4).

Notation

Throughout this thesis, we adopt the following notation: sets are denoted by curly braces, e.g., $\{a, b, c\}$; sequences by parentheses (a, a, b, c, a) ; continuous ranges by $[\alpha, \beta] := \{x \in \mathbb{R} \mid \alpha \leq x \leq \beta\}$ for $\alpha, \beta \in \mathbb{R}$; discrete integer ranges by $[1 : N] := \{1, 2, \dots, N\}$ for some $N \in \mathbb{N}$. Vectors are represented by lowercase letters (e.g., x), and matrices are denoted by uppercase letters (e.g., $\mathbf{X}$), with coefficients given by $\mathbf{X}(n, m)$ for $n, m \in \mathbb{N}$. Matrices are displayed in transposed form, with n denoting the horizontal index and m the vertical index. An overview of commonly used symbols and operators is provided on pages 188–189.

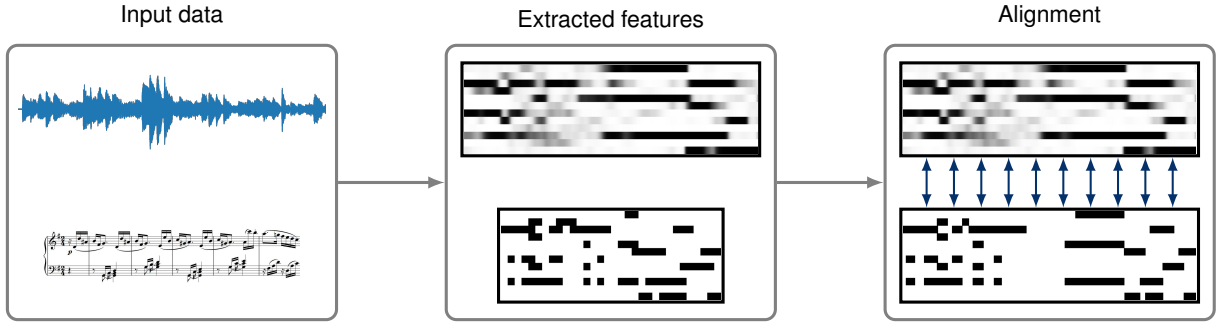


Figure 2.2: Overview of a typical two-stage music synchronization pipeline for score–audio synchronization. First, suitable features are extracted from the input data. Second, an alignment is computed between these features.

2.2 Feature Extraction and Representation

For synchronization tasks, feature representations must capture information that is relevant for the synchronization such as the harmonic, melodic, and rhythmic characteristics of a music recording. At the same time, features should remain robust to variations that are irrelevant such as variations in instrumentation, timbre, tuning, dynamics, and recording conditions. Typically, obtaining such representations involves two stages: first, extracting spectro-temporal features from the audio signal, and second, transforming these features into representations with desirable invariance properties, such as robustness to octave shifts or changes in instrumentation.

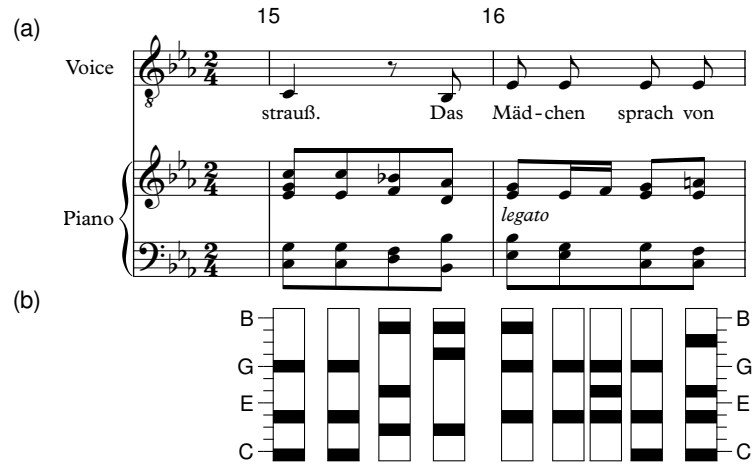
2.2.1 Feature Extraction

While features can be readily obtained from symbolic representations such as musical scores or MIDI files, they must be estimated when working with audio recordings. This estimation step is commonly addressed using either classical signal processing techniques or data-driven approaches based on DL. Throughout this thesis, we generally denote the two feature sequences used in a synchronization algorithm as $X = (x_1, \dots, x_N)$ and $Y = (y_1, \dots, y_M)$ with elements $x_n \in \mathcal{F}_X$ for $n \in [1 : N]$ and $y_m \in \mathcal{F}_Y$ for $m \in [1 : M]$, where $\mathcal{F}_X$ and $\mathcal{F}_Y$ denote suitably defined feature spaces.

Classical Signal Processing

Traditionally, many alignment pipelines rely on features derived using classical signal processing techniques, such as STFT- or MEL-based representations [98], the CQT [129], or filterbank-based approaches [44, 103]. These hand-crafted features are generally robust across different musical genres and acoustic conditions. However, their performance can degrade in complex musical scenarios characterized by high polyphony or instrumentation with strong overtone content.

Figure 2.3: (a) Musical score and (b) corresponding chroma features for an excerpt from the song “Gute Nacht” of the song cycle “Winterreise” by Franz Schubert.



Deep Learning

With recent advances in DL, numerous systems have been proposed for chroma estimation [80, 154], multi-pitch estimation [72, 80, 136, 137, 154, 158], and onset detection [18, 19, 113, 127]. Furthermore, the availability of large annotated datasets containing aligned audio and note labels, such as MAESTRO [64] and MusicNet [136], has enabled modern transcription systems to accurately estimate full sets of control signals such as note onsets and offsets, velocity and piano pedaling [49, 63, 65, 90]. While these DL-based feature extraction approaches achieve high accuracy in domains with abundant training data, such as piano music, their performance may degrade when applied to unseen acoustic conditions, instruments, or musical styles.

Overall, classical signal processing methods offer strong robustness and generalization, whereas DL approaches provide higher accuracy when sufficient training data are available. The choice of feature extraction method therefore depends on the target application and domain characteristics.

2.2.2 Chroma Features

Among the various feature representations used in MIR, chroma features are particularly well suited for synchronization tasks [98]. Chroma describes the perceived pitch class, or “color”, of a note and is defined over the set $\{C, C^\sharp, \dots, A^\sharp, B\}$. By design, chroma features are invariant with respect to octave changes, mapping notes of the same pitch class, e.g., C_3 and C_5 , to the same chroma bin C (see Figure 2.3 for an illustration). As a result, the twelve-dimensional chroma representation is well-suited for aligning different versions of the same musical work, such as recordings performed on instruments with different pitch ranges or timbral characteristics (e.g., harpsichord and grand piano), or for matching score representations to audio performances where octave doublings or register shifts may occur without altering the harmonic structure.

Formally, consider a feature sequence with semitone resolution along the frequency axis representing the 128 MIDI pitches, denoted by $X^{\text{pitch}} = (x_1^{\text{pitch}}, \dots, x_N^{\text{pitch}})$ with $x_n^{\text{pitch}} \in \mathbb{R}^{128}$. A chroma feature sequence $X^{\text{chroma}} = (x_1^{\text{chroma}}, \dots, x_N^{\text{chroma}})$ with $x_n^{\text{chroma}} \in \mathbb{R}^{12}$ can then be computed by aggregating values across octaves, for pitch bins corresponding to the same pitch classes [98]:

$$x_n^{\text{chroma}}(c) = \sum_{\{p \in [1:128] \mid p \bmod 12 = c-1\}} x_n^{\text{pitch}}(p), \quad \text{for } c \in [1 : 12]. \quad (2.1)$$

Due to their low dimensionality and octave invariance properties, chroma features form the basis of many synchronization approaches and serve as a standard feature representation throughout this thesis, both for audio and symbolic data.

2.3 Cost Measures

In the following section, we briefly introduce commonly used cost functions for DNN training and sequence alignment. Throughout, we denote predictions by x and labels by y .

Binary classification is the simplest classification setup, where a model predicts whether an input belongs to one of two classes. The prediction is denoted by a scalar $x \in [0, 1]$, representing the probability of the positive class and typically obtained via sigmoid activation, and the target is a binary label $y \in \{0, 1\}$. For this case, the Binary Cross-Entropy (BCE) cost $[0, 1] \times \{0, 1\} \rightarrow \mathbb{R}_+$ is defined as:

$$c_{\text{BCE}}(x, y) := -y \log x - (1 - y) \log(1 - x). \quad (2.2)$$

This formulation naturally extends to multi-label classification problems, where each input can belong to multiple classes and the predictions $x = [x^{(1)}, \dots, x^{(D)}]^\top \in [0, 1]^D$ (obtained via sigmoid activation) and targets $y = [y^{(1)}, \dots, y^{(D)}]^\top \in \{0, 1\}^D$ represent multiple class or symbol probabilities. In this case, the BCE loss function $c_{\text{BCE}} : [0, 1]^D \times \{0, 1\}^D \rightarrow \mathbb{R}_+$ is applied independently to each class and the sum is calculated [52]:

$$c_{\text{BCE}}(x, y) := - \sum_{d=1}^D \left(y^{(d)} \log x^{(d)} + (1 - y^{(d)}) \log(1 - x^{(d)}) \right). \quad (2.3)$$

In contrast, in the single-label case, where we assume that each input belongs to exactly one of D classes, the predictions $x \in \Delta^D$ represent a probability distribution over the classes. We then employ the categorical cross-entropy loss $c_{\text{CE}} : \Delta^D \times \mathcal{E}^D \rightarrow \mathbb{R}_+$, defined as:

$$c_{\text{CE}}(x, y) := - \sum_{d=1}^D y^{(d)} \log x^{(d)}, \quad (2.4)$$

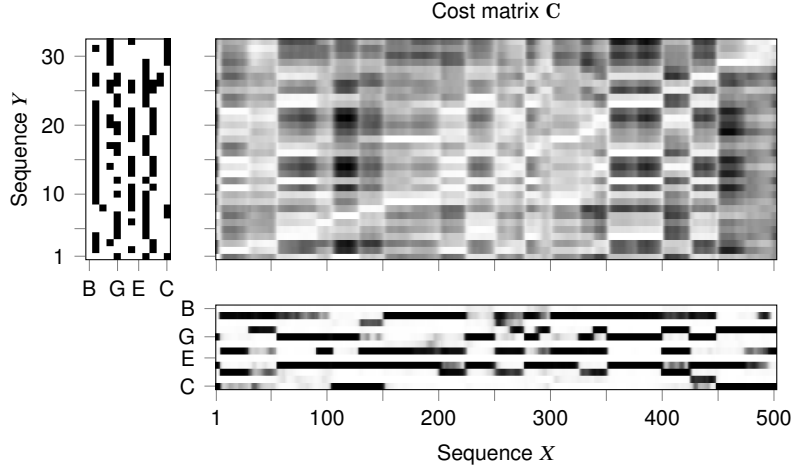


Figure 2.4: Illustration of two chroma sequences X and Y , and the resulting cost matrix C .

where $x \in \Delta^D$ is typically obtained via softmax activation, $\Delta^D = \{x \in [0, 1]^D \mid \sum_{d=1}^D x^{(d)} = 1\}$ is the D -dimensional probability simplex, and $\mathcal{E}^D = \{x \in \{0, 1\}^D \mid \sum_{d=1}^D x^{(d)} = 1\}$ is the set of standard basis vectors in $\mathbb{R}^D$.

For real-valued predictions and targets $x, y \in \mathbb{R}^D$, a common choice is the Mean Squared Error (MSE) cost function $c_{\text{MSE}} : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}_+$ [52]:

$$c_{\text{MSE}}(x, y) := \frac{1}{D} \sum_{d=1}^D (x^{(d)} - y^{(d)})^2. \quad (2.5)$$

Alternatively, the Cosine Distance $c_{\text{COS}} : \mathbb{R}^D \times \mathbb{R}^D \rightarrow [0, 2]$ can be used to assess the length-independent similarity between two vectors $x, y \in \mathbb{R}^D$ [98]:

$$c_{\text{COS}}(x, y) := 1 - \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}. \quad (2.6)$$

Here, the resulting value lies in $[0, 2]$, with smaller values indicating higher similarity. For non-negative $x, y \in \mathbb{R}_{\geq 0}$, the Cosine Distance is bounded in $[0, 1]$.

2.4 Dynamic Time Warping (DTW)

In this section, we briefly introduce the classical DTW algorithm as a tool for sequence alignment [98].

2.4.1 Sequences and Cost Matrix

Let $c : \mathcal{F}_X \times \mathcal{F}_Y \rightarrow \mathbb{R}$ be a local cost measure to compare two features $x \in \mathcal{F}_X$ and $y \in \mathcal{F}_Y$, as defined in Section 2.3. Then, we collect all pair-wise cost values between sequences $X \in \mathcal{F}_X^N$ and $Y \in \mathcal{F}_Y^M$ in a local cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ with entries:

$$\mathbf{C}(n, m) := c(x_n, y_m). \quad (2.7)$$

An illustration of two chroma sequences and the resulting pair-wise cost matrix is provided in Figure 2.4.

2.4.2 Alignment Paths

Alignments are defined in terms of alignment paths $P = (p_1, \dots, p_L)$ of length L , where each element $p_\ell = (n_\ell, m_\ell) \in [1 : N] \times [1 : M]$ denotes a pair of indices referring to the sequences X and Y . Equivalently, p_ℓ can be interpreted as an index to a cell $\mathbf{C}(n_\ell, m_\ell)$ of the local cost matrix $\mathbf{C}$. In classical DTW, a valid alignment path must satisfy boundary, monotonicity, and step-size constraints [98]. In the context of music synchronization, these constraints imply that the aligned sequences start and end at their respective first and last elements (boundary condition), that temporal order is preserved and sequences cannot be played backwards (monotonicity condition), and that musical events are matched without omissions (step-size condition).

First, the boundary constraint requires the alignment to start at the first elements of both sequences and to end at their last elements:

$$p_1 = (1, 1) \quad \text{and} \quad p_L = (N, M). \quad (2.8)$$

Second, the monotonicity constraint enforces a causal progression along both sequences, meaning that the alignment may only move forward in time:

$$n_{\ell-1} \leq n_\ell \quad \text{and} \quad m_{\ell-1} \leq m_\ell \quad \text{for} \quad \ell \in [2 : L]. \quad (2.9)$$

Finally, the step-size constraint restricts the set of admissible local moves. In classical DTW, only unit steps in the horizontal, vertical, or diagonal direction are allowed [97]:

$$p_\ell - p_{\ell-1} \in \{(1, 0), (0, 1), (1, 1)\} \quad \text{for} \quad \ell \in [2 : L]. \quad (2.10)$$

These constraints ensure that the alignment path proceeds continuously through both sequences while allowing local stretching and compression in time. Figure 2.5a illustrates an alignment path P between two sequences via a sequence of arrows, where each arrow corresponds to a tuple (n_ℓ, m_ℓ) . The path

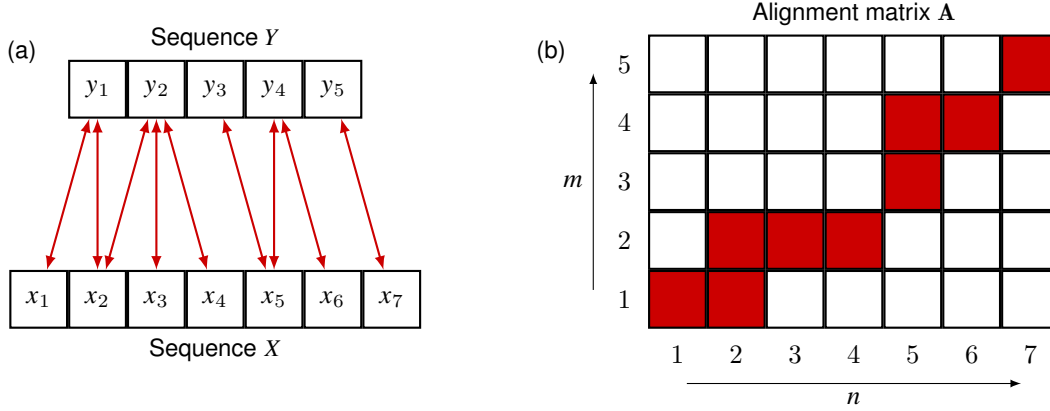


Figure 2.5: Illustration of different methods for alignment visualization. (a) Local pair-wise assignment of corresponding sequence elements. (b) Global path encoding in a binary alignment matrix.

length L is bounded by:

$$\max(N, M) \leq L \leq N + M - 1. \quad (2.11)$$

We denote by $\mathcal{P}$ the set of all alignment paths (for sequences of lengths N and M) that satisfy the boundary, monotonicity, and step-size constraints. The cardinality of $\mathcal{P}$ for given N and M under these constraints is given by the Delannoy formula $\mathcal{D}(N, M)$ [8]:

$$|\mathcal{P}| = \mathcal{D}(N, M) = \sum_{k=0}^{\min(N, M)} \binom{N}{k} \binom{M}{k} 2^k. \quad (2.12)$$

Alternatively, an alignment path P can be defined as a binary alignment matrix $\mathbf{A} \in \{0, 1\}^{N \times M}$ (see Figure 2.5b) with:

$$\mathbf{A}(n, m) = \begin{cases} 1, & (n, m) \in P, \\ 0, & (n, m) \notin P. \end{cases} \quad (2.13)$$

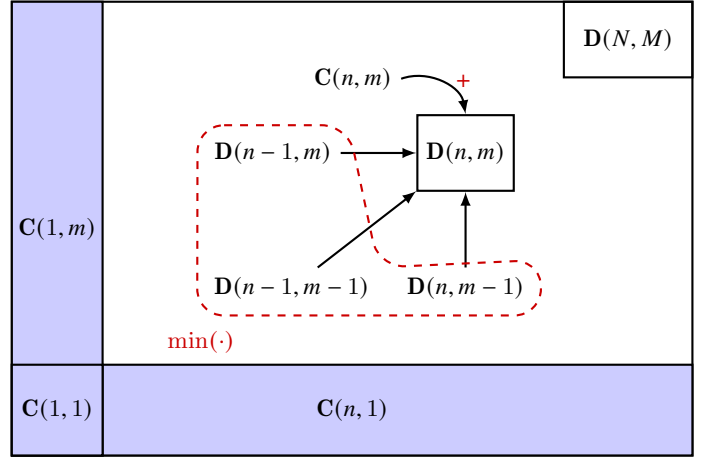
The set of all valid alignment matrices, i.e., matrices constructed from paths $P \in \mathcal{P}$ as given by (2.13) is denoted as $\mathcal{A}$.

2.4.3 DTW Objective

The cost $c_P(\mathbf{C})$ of aligning sequences X and Y with pair-wise cost stored in a local cost matrix $\mathbf{C}$ under the path P is calculated by summing up the pairwise cost values of all aligned elements:

$$c_P(\mathbf{C}) := \sum_{(n, m) \in P} \mathbf{C}(n, m). \quad (2.14)$$

Figure 2.6: Schematic overview of the DTW forward pass.



Equivalently, using the alignment definition via alignment matrices $\mathbf{A}$, the cost of one alignment $c_{\mathbf{A}}(\mathbf{C})$ is

$$c_{\mathbf{A}}(\mathbf{C}) := \langle \mathbf{A}, \mathbf{C} \rangle, \quad (2.15)$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. The DTW cost $\text{DTW}(\mathbf{C})$ is defined as the minimum alignment cost over $\mathbf{C}$ and can be obtained by full search over all allowed paths:

$$\text{DTW}(\mathbf{C}) := \min_{P \in \mathcal{P}} c_P(\mathbf{C}) = \min_{\mathbf{A} \in \mathcal{A}} c_{\mathbf{A}}(\mathbf{C}). \quad (2.16)$$

If the goal is not to obtain the minimum alignment cost but the optimal alignment path P^* or $\mathbf{A}^*$ (which is not necessarily unique), it can be recovered as the argmin of (2.16):

$$P^* = \arg \min_{P \in \mathcal{P}} c_P(\mathbf{C}) \quad (2.17)$$

$$\mathbf{A}^* = \arg \min_{\mathbf{A} \in \mathcal{A}} c_{\mathbf{A}}(\mathbf{C}). \quad (2.18)$$

2.4.4 Dynamic Programming

While the evaluation of the DTW cost and optimum path from (2.16) to (2.18) is theoretically possible, the number of allowed alignment paths $|\mathcal{P}|$ grows exponentially with the sequence lengths, rendering a direct computation infeasible in practice. To address this, a DP algorithm is used to compute the DTW cost (forward pass) and optimum path (backward pass). For a concise description of the optimality of the DP recursions in DTW-like problems, we refer to Section 10.3.

DTW Forward Recursion

Using Bellman's principle of optimality [11], which describes solving a larger problem via a series of simpler subproblems, the DTW cost is computed recursively by iteratively filling an accumulated cost

Chapter 2. Background: Music Synchronization

matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$, where

$$\mathbf{D}(n, m) := \text{DTW}(\mathbf{C}(1:n, 1:m)) \quad (2.19)$$

denotes the minimum alignment cost for the subsequences $X(1 : n)$ and $Y(1 : m)$ [98]. The recursion is initialized at the first row and first column of the matrix as

$$\mathbf{D}(n, 1) = \sum_{i=1}^n \mathbf{C}(i, 1) \quad \text{for } n \in [1 : N] \quad (2.20)$$

$$\mathbf{D}(1, m) = \sum_{j=1}^m \mathbf{C}(1, j) \quad \text{for } m \in [1 : M], \quad (2.21)$$

and is then computed recursively via

$$\mathbf{D}(n, m) = \mathbf{C}(n, m) + \min(\mathbf{D}(n-1, m), \mathbf{D}(n, m-1), \mathbf{D}(n-1, m-1)) \quad (2.22)$$

for $n \in [2 : N]$ and $m \in [2 : M]$ [98] (see Figure 2.6). The total DTW cost is obtained from the last entry of the accumulated cost matrix:

$$\text{DTW}(\mathbf{C}) = \mathbf{D}(N, M). \quad (2.23)$$

DTW Backward Recursion

To recover the optimal alignment path P^* , a backtracking procedure is applied [98]. Let $q_1 = (N, M)$ denote the last element of the optimal path. For each alignment step $q_\ell = (n, m)$, the predecessor is chosen as

$$q_{\ell+1} = \arg \min_{(n', m') \in \tilde{q}_{n, m}} \mathbf{D}(n', m'), \quad (2.24)$$

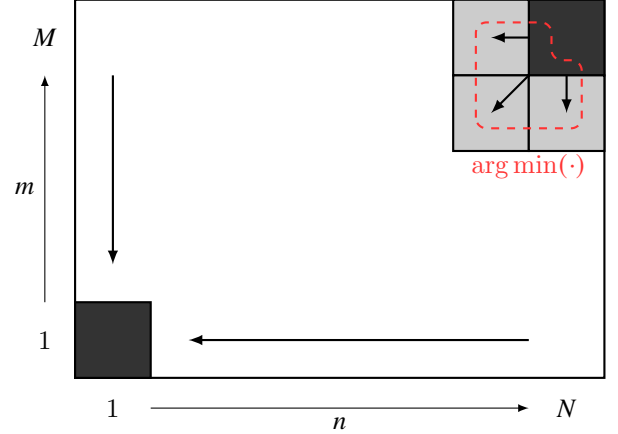
where $\tilde{q}_{n, m} = \{(n-1, m), (n, m-1), (n-1, m-1)\} \cap [1 : N] \times [1 : M]$ contains the admissible prior positions, considering only valid in-bound indices (see Figure 2.7). If a boundary is reached, i.e., $n = 1$ or $m = 1$, backtracking continues along the border:

$$q_{\ell+1} = (1, m-1) \quad \text{if } n = 1, \quad (2.25)$$

$$q_{\ell+1} = (n-1, 1) \quad \text{if } m = 1. \quad (2.26)$$

The optimal alignment path then follows as $P^* = (q_L, \dots, q_1)$, where $L = \ell$ after the final recursion step [98]. The complexity of both the forward and backward passes is $\mathcal{O}(NM)$ in time and space.

Figure 2.7: Schematic overview of the DTW backward pass.



2.5 Robust, Efficient, and High-Accuracy Synchronization

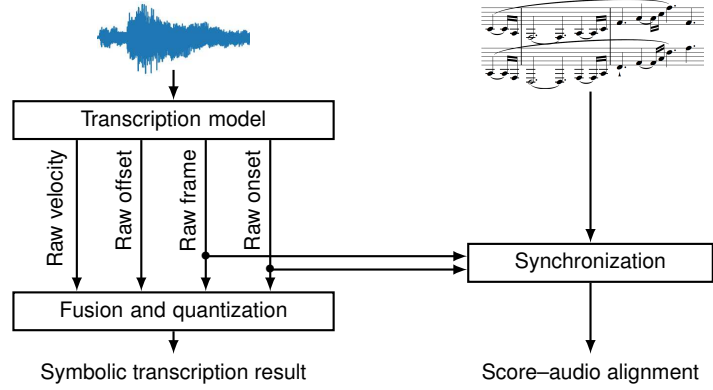
Going beyond classical DTW, numerous extensions and regularizations have been proposed with the goal of improving robustness, computational efficiency, and alignment accuracy. From the perspective of robustness, variants of DTW introduced modified step-size patterns (e.g., allowing 2–1 steps) and local weighting schemes with individual weights for the vertical, horizontal, and diagonal alignment steps [98]. Further robustness can be achieved by constraining the admissible alignment paths to predefined regions, such as the Sakoe–Chiba band [125] or the Itakura parallelogram [70], which limit excessive warping. Another important strategy is the use of multi-scale approaches [100, 126], in which alignment is performed from coarse to fine temporal resolutions, thereby stabilizing the path estimation and reducing the impact of local noise. Furthermore, certain anchor points can be annotated manually to restrict the alignment to defined regions, enabling robust synchronization even in challenging musical scenarios [103, 155].

Additional speed-ups can be obtained by restricting the alignment computation to local constraint regions [87], thereby avoiding evaluations in low-probability or infeasible areas. These ideas can also be combined, for example, in memory-restricted multi-scale DTW formulations [120], which jointly address the aforementioned strategies, thereby improving runtime and memory efficiency, and enabling the processing of long sequences.

When synchronizing recordings with highly polyphonic content, musical events that are notated as simultaneous may be distributed across multiple time frames in performances, for example due to arpeggiation or slight timing imprecision. After an initial global sequence alignment, these fine-grained note-level timing discrepancies can be compensated using local matching techniques [90, 110].

Finally, improvements in synchronization accuracy can be achieved by enhancing the underlying feature representations. A common strategy is to combine complementary features, such as activation-based (stability-oriented) features with onset-based (precision-oriented) features [44]. Such features can be extracted using a variety of methods, including modern learning-based onset detection techniques, which have been shown to further improve alignment accuracy [113].

Figure 2.8: Schematic overview of the proposed audio–score synchronization pipeline using raw features from a transcription model.



2.6 Experiment: Music Synchronization with Neural Features

In this section, which is based on [169], we conduct a first music synchronization experiment, highlighting the importance of suitable feature estimators and representations. In particular, we compare features extracted from transcription models to signal processing-based features as input to music alignment algorithms (see Figure 2.8). We conclude with insights into methods for evaluating synchronization accuracy in scenarios where reference annotations are not available.

2.6.1 Synchronization Pipeline

In this section, we provide an overview of the synchronization pipeline. First, we describe the feature extractors and types of feature representations, before we outline the DTW-based alignment step. An overview of all elements in the pipeline and their abbreviations is provided in Table 2.1. Following the notation in [63], we distinguish between two types of features: *frame* features encode when a note is active (in the piano case, this corresponds to the time until a key is released, or until the sustain phase ends), and *onset* features encode only the beginning of a note (in the piano case when a key is pressed).

Feature Extraction Model

Filterbank. Before the advent of today’s DL-based feature extraction models, traditional signal processing techniques were a common way to extract features from audio recordings. For example, the standard implementation of Sync Toolbox [103] uses a multirate filterbank (FB) to estimate frame-wise note activity and onsets.

Transcription Model. In recent years, AMT systems based on DL have shown significant improvements in performance [12]. One of the groundbreaking architectures is the *Onsets and Frames* architecture by Hawthorne et al. [63], which has separate prediction heads to estimate onsets and frames. Maman et al. [90] proposed a strategy to train a model based on the onsets and frames architecture on diverse

2.6. Experiment: Music Synchronization with Neural Features

Table 2.1: Overview of short notation for all components of the processing chain.

Feature Extraction Model	
FB	Filterbank
T1	Onsets and frames transcription model [90]
T2	High-resolution transcription model [77]
DK	Disklavier
Feature Representation	
RAW	Continuous pitch and onset probabilities
SYM	Thresholded and discretized pitches and onsets
Alignment Technique	
O	Onset features using standard DTW
OF	Onset and frame features using MrMsDTW from [103]

and unaligned pairs of audio data and musical scores. This has led to improved performance on unseen datasets, generalizing across instrumentations, acoustic conditions, and styles. In the following, we refer to the transcription model from [90], trained on MusicNet [136] with re-aligned labels, as T1. Kong et al. [77] extend the onsets and frames architecture by additionally modeling sustain pedal activity, therefore providing more robust training in the presence of misaligned offset information. We refer to the transcription system from [77], trained on MAESTRO [64], as T2.¹

Disklavier. Certain datasets such as MAESTRO [64] and ASAP [47] include pairs of audio recordings and reference note information by having the pieces performed on a Disklavier.² We refer to features directly extracted from the symbolic Disklavier track as DK, and use them as an upper bound for the prediction accuracy of a learning-based feature extractor.

Feature Representation

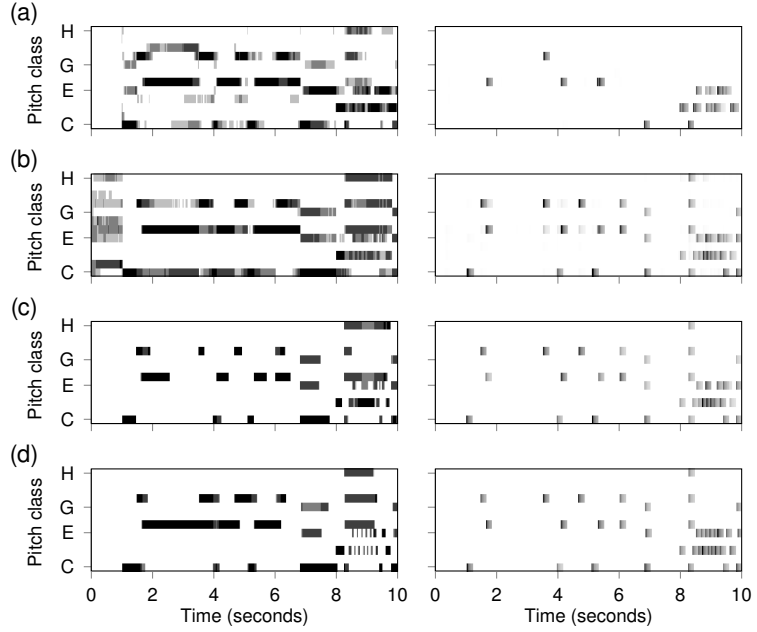
Raw features. For each sequence of input audio, the feature extractors FB, T1, and T2 predict continuous pitch- and frame-wise probabilities $\mathbf{p}_{\text{raw}}^{\text{frame}}, \mathbf{p}_{\text{raw}}^{\text{onset}} \in [0, 1]^{88 \times N}$ for frame activity and note onset, respectively. These feature matrices can be thought of as RAW features and are commonly stored in a pianoroll-like representation for 88 pitches and N time frames (T1 and T2 additionally predict such probabilities for note velocity and offset). Figure 2.9a/b illustrates RAW features for a piece from the ASAP dataset [47], computed by the FB and T1 feature extractors.

Symbolic features. In AMT, the raw predictions for frames, onsets, and offsets are fused and quantized in a postprocessing step, yielding binary estimates about which keys have been pressed. In particular, note sustain (frame activity) is conditioned on a previously occurring note onset [63, 77]. This postprocessing step outputs a sequence of symbolic control messages for note onset and offset events, with additional control messages for pedal information in the case of T2. We denote these binary and symbolic-like

¹ Note that we do not include transcription models that directly output a tokenized sequence of MIDI messages (where we cannot access raw pitch probabilities), such as the MT3 model by Hawthorne et al. [65].

² A Disklavier is an acoustic piano equipped with sensors that capture key and pedal events and store them as symbolic MIDI files.

Figure 2.9: Chroma (left) and DLNCO (right) features for Beethoven’s Appassionata (Op. 57-1) played by F. Dupree. (a) FB_RAW. (b) T1_RAW. (c) T1_SYM. (d) DK_SYM.



features as SYM features and, for usage in our synchronization pipeline, store these in the form of two discretized pianorolls $\mathbf{P}_{\text{sym}}^{\text{frame}}, \mathbf{P}_{\text{sym}}^{\text{onset}} \in \{0, 1\}^{88 \times N}$ for frame activity and onset events, respectively. Note that in the case of Disklavier (DK), no RAW features are available and thus only SYM features are used. Figure 2.9c/d illustrates SYM features for T1 and DK.

Comparison. In Figure 2.9 we qualitatively compare RAW features from FB and T1 as well as SYM features from T1 to the DK reference. While FB_RAW in Figure 2.9a shows many false positive chroma events and misses many onsets compared to DK in Figure 2.9d, the chroma features of T1_RAW in Figure 2.9b are relatively stable and onsets perfectly coincide with DK. Thresholding the RAW transcription results to T1_SYM features (Figure 2.9c) yields varying and often shortened note durations in the chroma representation compared to DK, indicating possible instabilities when using these features for computing an alignment.

Alignment Technique

We use two variants of DTW to compute the optimal alignment between two feature sequences.

Onset features. As a first approach and in line with previous work [90, 121], we use only onset features and convert them to a twelve-dimensional pitch class representation. Using the Euclidean distance function, we compute the cost matrix between the onset feature sequences of the two versions to be aligned. We use standard DTW with unit steps in the horizontal, vertical, and diagonal direction with step weights (1.5, 1.5, 2) to compute the minimum cost path between the two sequences [98]. We refer to this approach, using only onset features, as O.

2.6. Experiment: Music Synchronization with Neural Features

Onset and frame features. As a second alignment variant, we choose a high-resolution approach [44] that combines frame and onset features. Using frame features yields robustness on the coarse temporal level, while onset features provide precision on the fine level by precisely aligning note onsets [113]. In this approach, we again convert frame and onset features into pitch class representations and additionally add a decay to the onset features. We refer to [44] for a description of these Decaying Locally Normalized Chroma Onset (DLNCO) features. Next, we compute separate cost matrices for frame features (using the cosine distance) and for onset features (using the Euclidean distance). Afterward, we add the two cost matrices for frame and onset features and use DTW with step weights (1.5, 1.5, 2) to compute the optimum alignment path on the combined cost matrix. We refer to [103, 120] for an efficient multi-resolution and multi-scale implementation of DTW. We denote the described approach, using a combination of onset and frame features, as OF. Note that we do not consider using only frame features (commonly called chroma features), as previous work has shown a lack of precision in this case. For example, Ewert et al. observe a 100 % increase of the alignment error when using frame features instead of combined frame and onset features for the case of piano music, where onsets are clearly defined [44].

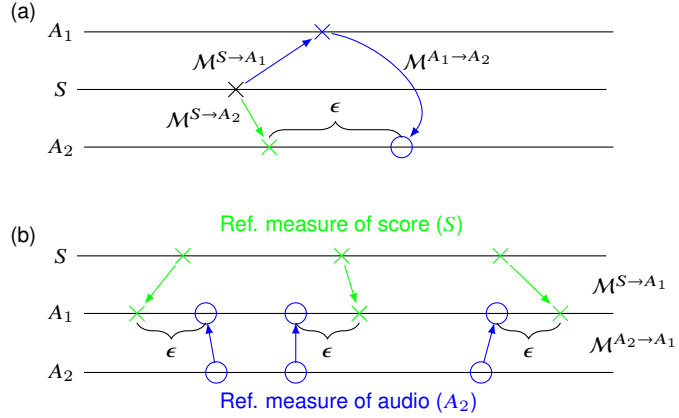
2.6.2 Dataset: ASAP

For our experiments, we require pairs of high-quality audio recordings, precise MIDI performance data (i.e., Disklavier recordings), and accurate symbolic note information. The MAESTRO dataset [64] fulfills the first two requirements, as it contains studio-quality recordings captured with Disklavier pianos. The ASAP dataset [47, 117] extends MAESTRO by pairing these performances with aligned symbolic scores.

To ensure the availability of reliable and manually curated score material, we restrict our analysis to recordings of Beethoven’s piano sonatas contained in ASAP. For these works, high-quality symbolic scores are also available through our BPSD [170] (see Chapter 3), which provides carefully curated digital editions of the corresponding sheet music. Using ASAP allows us to easily identify the relevant performances and verify their musical structure (matching to the BPSD), while the BPSD provides the accurate symbolic representation required for our experiments. The resulting subset of ASAP comprises 13 recordings with a total duration of 103 min.

A limitation of the ASAP/MAESTRO recordings is that they were captured exclusively with Disklavier instruments in recording environments with similar acoustic properties. While this setup yields highly accurate symbolic data, the resulting audio recordings may not fully reflect the variability of other recording scenarios, such as live performances or historical recordings.

Figure 2.10: Schematic illustration of (a) note onset transfer and (b) measure transfer heuristics.



2.6.3 Quantifying Synchronization Accuracy

In this section, we describe the heuristics used to assess the accuracy of our score–audio and audio–audio synchronization pipelines. We refer to [119] for a detailed discussion about the analysis of synchronization accuracy when ground-truth annotations are not available.

Notation

We first introduce some notation for aligning time points between two different versions V_1 and V_2 of a piece. We assume that these versions have continuous time axes $[0, T_1]$ and $[0, T_2]$, which can either be in physical time (for audio recordings, in seconds) or in musical time (for score-related data, in measures). From the alignment algorithms described in Section 2.6.1, we obtain a monotonic mapping function $\mathcal{M}^{V_1 \rightarrow V_2} : [0, T_1] \rightarrow [0, T_2]$ to transfer time instants from the timeline of one version to the other. Note that even though the alignment result obtained from DTW maps discrete time axes, our assumption of having continuous time axes can be obtained by using suitable interpolation techniques, see [119].

Accuracy Evaluation without Reference Annotations: Note Onset Transfer

In order to evaluate score–audio synchronization on datasets without time-instance annotations for audio recordings, we use an approach that transfers note onsets, illustrated in Figure 2.10a, to assess the synchronization accuracy. We consider three versions: $V_1 = S$ being a score, and $V_2 = A_1$ and $V_3 = A_2$ being different audio versions of the same piece.

First, we obtain audio features for two different audio recordings A_1, A_2 of the same piece, as well as features for the score S . For each version, we perform score–audio synchronization to obtain alignment functions $\mathcal{M}^{S \rightarrow A_1}$ and $\mathcal{M}^{S \rightarrow A_2}$ from musical to physical time. Using these mapping functions, we map note event onsets $t_1^S, \dots, t_M^S$ (where M is the number of onsets), given in musical time from the score to the physical time of the audio recordings. In a second step, we transfer the aligned onset positions from

2.6. Experiment: Music Synchronization with Neural Features

the first to the second audio using audio–audio synchronization via the mapping function $\mathcal{M}^{A_1 \rightarrow A_2}$ and compute the absolute error

$$\epsilon_m = \left| \mathcal{M}^{S \rightarrow A_2}(t_m^S) - \mathcal{M}^{A_1 \rightarrow A_2} \left(\mathcal{M}^{S \rightarrow A_1}(t_m^S) \right) \right| \quad (2.27)$$

between these transferred time points and the ones obtained from score–audio synchronization. We then compute the mean, median, and percentiles (i.e., the 90th and 95th percentiles of our error distribution) of the errors ϵ_m for all $m \in [1 : M]$.

Accuracy Evaluation with Reference Annotations: Measure Transfer

We next describe a method for evaluating score–audio synchronization if there are time instance annotations available for at least one audio version. We again consider three versions: $V_1 = S$ being a score, and $V_2 = A_1$ and $V_3 = A_2$ being different audio versions of the same piece. We choose A_2 to be a recording for which we have access to manually annotated measure time points $t_1^{A_2}, \dots, t_M^{A_2}$, for example, measure positions. Using audio–audio synchronization, we obtain a mapping $\mathcal{M}^{A_2 \rightarrow A_1}$ to transfer these measure positions to the first audio version A_1 . Similarly, we transfer measure positions $t_1^S, \dots, t_M^S$ obtained from the score S to the first audio A_1 using a mapping $\mathcal{M}^{S \rightarrow A_1}$. In a last step, as illustrated in Figure 2.10b, we calculate the absolute error of the time positions transferred from S and A_2 :

$$\epsilon_m = \left| \mathcal{M}^{S \rightarrow A_1}(t_m^S) - \mathcal{M}^{A_2 \rightarrow A_1}(t_m^{A_2}) \right|, \quad (2.28)$$

and compute statistics over ϵ_m for all $m \in [1 : M]$. While we cannot apply this heuristic for the ASAP dataset due to a lack of reference annotations, we already introduce it here for later use in synchronization evaluation on the BPSD (see Section 3.4).

In both heuristics, we assume that the synchronization accuracy is high if the time instances transferred via two different branches have small deviations. Note that this is only a *necessary* and not a *sufficient* condition for alignment quality; For example, one can construct a simple linear warping that results in a note onset transfer error of zero. Nevertheless, in practice these metrics give good indicators of the alignment performance (see also [119]).

2.6.4 Evaluation

In this section, we evaluate synchronization accuracy on the ASAP dataset using our note onset transfer approach described in Section 2.6.3. For each pair of audio files, we calculate the mean, median, 90 and 95 percentiles of the absolute alignment error in ms, and report averaged results in Table 2.2. For an evaluation on the BPSD, we refer to Chapter 3.

Table 2.2: ASAP: Absolute error in milliseconds for note onset transfer heuristic.

Feature	Mean	Median	90th	95th
T1_SYM_O	89	0	281	467
T1_SYM_OF	66	12	153	293
T1_RAW_OF	29	8	64	146
T2_SYM_O	31	0	102	192
T2_SYM_OF	22	2	49	111
T2_RAW_OF	21	7	43	99
DK_SYM_O	37	0	130	216
DK_SYM_OF	25	2	57	123
FB_RAW_OF	64	20	146	268

Analyzing the median absolute error, which we consider an indicator for the achievable accuracy under average conditions, we find perfect alignments ($\epsilon = 0$ ms) for at least 50 % of all note onsets when using only symbolic onset features (SYM_O) for the transcription models T1 and T2, as well as the Disklavier DK. To assess the methods’ robustness and the severeness of outliers, we next investigate the 95 % quantiles of the absolute alignment error. Using only symbolic onset features and standard DTW (SYM_O) yields the highest errors for all T1 (467 ms), T2 (192 ms), and DK (216 ms) variants, indicating a lack of robustness despite excellent median accuracy.

In the next step, we jointly use the frame and onset information from the symbolic features (SYM_OF) and observe a slight rise in the median error to 12 ms for T1, and to 2 ms for T2 and DK, respectively. While this indicates that the best achievable precision slightly deteriorates, we monitor a significant reduction of the 90th and 95th percentiles by approximately 50 % for all SYM_OF variants, indicating a vastly improved robustness towards outliers when combining frame and onset features in the computation of the alignment.

Lastly, we directly use the intermediate predictions for frames and onsets (RAW_OF) in our alignment pipeline. While the median absolute error is comparable to the one based on symbolic features, we again observe a significant decrease in the 90th and 95th percentiles. Using RAW_OF features in the T2 transcriber yields the lowest mean (21 ms) and percentiles (43 and 99 ms), even outperforming the usage of reference note information obtained from the Disklavier.³ We illustrate this finding with the intuitive example of a chord where notes are not played simultaneously, either due to a playing mistake or as a stylistic element, leading to a deviation of symbolic and actually performed note order. While the DK features strictly assign each note onset to one particular time frame and thus cause alignment instabilities in the given example, the continuous RAW predictions can smoothly cover neighboring time frames and thus allow for a robust alignment.

³ We note that the T2 model was trained on MAESTRO [64], which is the basis of ASAP [47]. Therefore, a separation for train and test data is not guaranteed for T2 in the experiments on ASAP. However, the DK features nevertheless are the upper limit of the achievable transcription accuracy.

3 Beethoven Piano Sonata Dataset: Alignment-Based Dataset Curation

This chapter is mainly based on [170]. Compiling the BPSD involved an extensive workflow, encompassing the preparation and processing of raw data, typesetting, manual corrections, annotation creation and transfer, as well as the utilization of the data for research. This complex process underwent multiple cycles of revision and correction, requiring a collaborative effort from a considerable number of individuals within the research team. All co-authors of the paper [170] actively participated in both the data curation process and the preparation of the final manuscript. On the technical level, Johannes Zeitler is the main contributor to the design, implementation, and evaluation of the alignment strategy, the consolidation of the annotations, and the implementation of the application-oriented case studies. Furthermore, Section 3.4 is based on [169]. Johannes Zeitler is the main contributor to this article, responsible for code implementation, evaluation, experiments, analysis, and writing. Ben Maman trained models used for evaluation. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

Large-scale MIR datasets encompassing multiple versions and modalities are most useful when all representations of a piece follow a coherent timeline, but such consistency is often missing in practice. This chapter presents the Beethoven Piano Sonata Dataset (BPSD), a carefully curated multi-version dataset that consolidates symbolic score representations, audio recordings of multiple performances, and annotations on a unified and coherent time axis.

A main factor enabling this temporal coherency is the synchronization pipeline introduced in the previous chapter, which integrates neural features into a DTW-based synchronization framework to align the different sources on a shared timeline. We evaluate this hybrid synchronization approach in the context of the BPSD curation, demonstrating improved robustness and accuracy across a diverse set of piano audio versions, including historical and low-quality recordings.

We then demonstrate the utility of the BPSD in representative application scenarios, including transcription, harmony analysis, and cross-version evaluation. Overall, with the BPSD we demonstrate the usage of robust synchronization as an enabling factor for alignment-based dataset design, transforming heterogeneous musical sources into a coherent experimental setup suitable for DNN training and evaluation. This practical scenario motivates the shift of perspective towards learning directly from weakly aligned data in the subsequent chapters.

This chapter is organized as follows. In Sections 3.1 and 3.2, we introduce the dataset and discuss related work with a particular focus on other resources conceptually or thematically associated with the BPSD. Section 3.3 outlines the overall organization of the BPSD and introduces its primary musical material. Section 3.4 presents an enhanced music synchronization approach used as our central method for aligning different versions and transferring annotations between them. Section 3.5 delves into the annotations and their main properties. Furthermore, Section 3.6 explores the potential of the BPSD by discussing three concrete application scenarios, and Section 3.7 concludes the chapter by suggesting possible research directions. Readers merely interested in differentiable alignment as a training objective in end-to-end learning pipelines may skip this chapter and proceed directly with Chapter 4.

3.1 Introduction

The rise of digital technology has brought about significant developments in computer science, recently with a remarkable success of data-driven methods utilizing deep learning (DL) techniques. This progress has reinforced the importance of comprehensive, systematic, and reliable datasets. In the field of Music Information Retrieval (MIR), carefully curated datasets have become indispensable resources, playing a crucial role in advancing tasks such as harmony analysis [116, 152], beat and downbeat estimation [20], structure analysis [111], and music transcription [12]. From a technical perspective, high-quality and well-controlled datasets are essential for evaluating and understanding DL methods, providing insights into their robustness, generalization capabilities, and uncovering potential data biases. For an illustrative study in this direction, particularly for the task of local key estimation, we refer the reader to [152]. Beyond the development and evaluation of analysis methods, such datasets enable musicological corpus studies, which allow for systematic analyses and explorations across entire corpora, thus going beyond individual pieces while contributing to an objective methodology for musicological research [93, 107, 131].

As a novel resource for such endeavors, we introduce the Beethoven Piano Sonata Dataset (BPSD), a multi-version corpus focusing on the first movements of Ludwig van Beethoven’s 32 piano sonatas.⁴ These works rank among the most popular pieces in the classical music repertoire, holding a special place in Western cultural history [28] and being performed and adapted innumerable times. Beethoven’s piano sonatas expand, advance, and further develop the traditional sonata form [68], explore innovative

⁴ This selection follows the convention of most editions and CD compilations to not include the early sonatas WoO 47 (*Kurfürstensonaten*), which have been subject to a computational study by Klauk et al. [75].

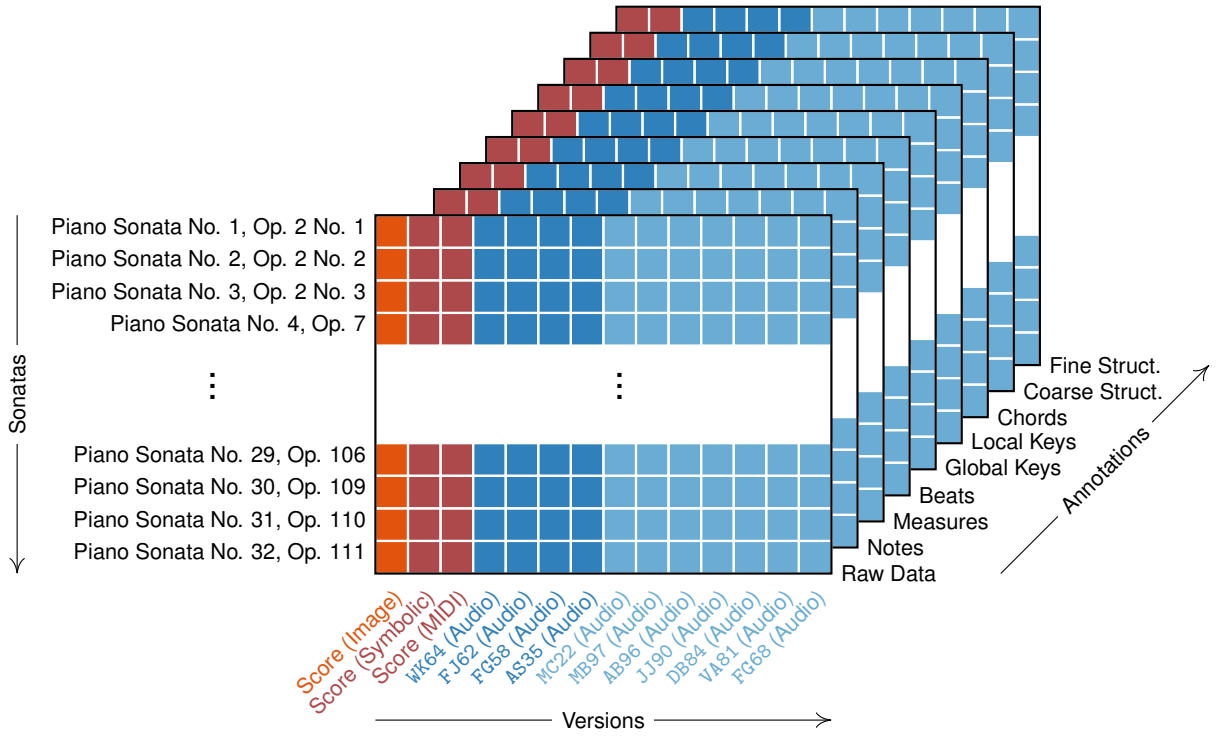


Figure 3.1: Schematic overview of the BPSD. For the first movements of all 32 piano sonatas, the dataset comprises raw data in different representations (*versions*) such as score images, symbolic score representations, and audio recordings of different performances. For the different versions, we provide time-aligned annotations of measure positions, beats, global and local keys, chords, and structural elements. The score and the four audio versions indicated in dark blue are in the public domain (EU).

harmonic progressions [32], and showcase an expressive range and dramatic intensity that elevated the genre and heavily influenced later composers [140]. Our BPSD contributes valuable resources on a work cycle of high musicological relevance, collecting, unifying, and providing a rich set of musical data and annotations. Regarding the primary material (*raw data*), the BPSD includes diverse representations of the sonata movements across various modalities. This encompasses sheet music in different machine-readable formats and audio recordings from eleven performances, with four of them being in the public domain (in the EU) and freely accessible for research purposes (see Figure 3.1 for an overview). Beyond the raw data, our dataset comprises secondary material in the form of carefully curated and linked *annotations*, encompassing measures, beats, note events, global and local keys, absolute and relative chords, and structural elements. Starting with annotations specified on a musical timeline (given in measures) based on sheet music, we employed semi-automated approaches that leverage music synchronization techniques to transfer these annotations to the physical timeline of the recordings (given in seconds). As a result, BPSD includes version-specific annotations for both the sheet music and the eleven audio recordings, forming a systematic and complete three-dimensional data tensor, as illustrated in Figure 3.1.

Regarding its practical applicability in a computational context, a key feature of the BPSD is its *coherence*, evident not only in the various annotations provided for all versions but also in its unifying approach to musical timeline and structures. Note that depending on the representation and performance, versions of

the same piece of music may exhibit different structures, resulting in significant temporal inconsistencies. For example, in sheet music, repeats are notated using repeat signs, possibly with alternative first and second endings. While the enumeration of measures involves assigning a unique number to each measure in sequential order, this order is violated when there are repeats, complicating matters from an algorithmic perspective. Furthermore, different performances may add or omit (e.g., playing the exposition of a sonata form only once) repetitions. Even worse, particularly in older live recordings, there may be severe playing errors, introducing and leaving out several measures.

To address these issues, we ensure temporal coherence in the BPSD by selecting a reference version for all movements and carefully adjusting musical timelines. In particular, we unfold repetitions in musical scores, enumerate measures contiguously, and edit audio recordings by suitably copying or deleting certain parts or sections. This process results in all versions adhering to the same musical timeline, each measure having a unique identifier, and alignments becoming well-defined without gaps across versions. This uniformity greatly enhances the usability of the BPSD for computational approaches, cross-version evaluation, visualization, navigation, and other applications. The dataset is accessible through a version-controlled repository on the Zenodo platform, and its DOI is 10.5281/zenodo.10847702.⁵

3.2 Related Datasets and Resources

The MIR community has provided a diverse range of research datasets, which constitute an important basis for advancing music-related computational research. Specifically, in the realm of data-driven methods, datasets play a crucial role as an essential component for training and evaluating DL models in MIR research. The RWC music database [7, 56] stands out as one of the first larger music datasets designed specifically for research purposes. It encompasses various genres, including popular, jazz, and classical music, providing both audio recordings and symbolic MIDI representations synchronized through the production processes or the application of alignment techniques. More recent examples include the multitrack dataset MedleyDB [13] designed for source separation and other MIR applications and the Erkomaishvili dataset [124] suitable for ethnomusicological research, both featuring high-quality annotations. In his thoughtful essay, Serra [131] makes a distinction between unstructured yet annotated *datasets* (or *test corpora*) and curated *research corpora*, identifying five essential criteria (purpose, coverage, completeness, quality, reusability) for a research corpus. The BPSD, along with several others mentioned in this chapter, can be considered a research corpus based on these criteria. For a comprehensive overview of diverse publicly accessible datasets in the field of MIR, we refer to [15]⁶.

In the subsequent discussion, we focus on datasets centered around piano music, many of which are specifically tailored to automatic music transcription (AMT), see, e.g., [12]. These datasets provide audio recordings of musical pieces along with corresponding symbolic encodings that are synchronized

⁵ <https://doi.org/10.5281/zenodo.10847702>

⁶ <https://mirdata.readthedocs.io/en/latest/>

3.2. Related Datasets and Resources

No.	Work ID	Name	Key	Mean Dur.	Min. Dur.	Max. Dur	Meas.	Structure
01	Op002No1-01		F:min	03:47	03:22 (AS35)	04:33 (WK64)	200	E-E-D-R
02	Op002No2-01		A:maj	07:04	06:23 (FG67)	07:45 (MC22)	452	E-E-D-R
03	Op002No3-01		C:maj	10:15	09:47 (FG58)	11:25 (MC22)	347	E-E-D-R-C
04	Op007-01	Grand Sonata	Eb:maj	08:17	07:27 (AS35)	08:58 (MC22)	497	E-E-D-R-C
05	Op010No1-01		C:min	05:33	04:41 (AS35)	06:13 (MC22)	388	E-E-D-R
06	Op010No2-01		F:maj	05:38	05:03 (FG67)	06:14 (VA81)	268	E-E-D-R
07	Op010No3-01		D:maj	06:59	06:26 (FJ62)	07:53 (JJ90)	467	E-E-D-R-C
08	Op013-01	Pathétique	C:min	08:56	08:06 (FG58)	09:57 (MC22)	431	I-E-E-D-R-C
09	Op014No1-01		E:maj	06:35	05:31 (VA81)	07:25 (AB96)	222	E-E-D-R-C
10	Op014No2-01		G:maj	07:06	05:49 (AS35)	07:56 (AB96)	263	E-E-D-R-C
11	Op022-01		Bb:maj	07:26	06:43 (AS35)	08:36 (MC22)	267	E-E-D-R
12	Op026-01	Funeral March	Ab:maj	08:01	06:51 (FG67)	10:02 (AS35)	219	T-V1-V2-V3-V4-V5
13	Op027No1-01	Son. q. u. fant.	Eb:maj	05:12	04:36 (AB96)	05:42 (FG58)	106	An-Al-T1
14	Op027No2-01	Moonlight	C#:min	06:01	04:58 (AS35)	07:28 (FG58)	69	P1-P2-P3-C
15	Op028-01	Pastoral	D:maj	09:58	08:58 (FJ62)	11:39 (MC22)	622	E-E-D-R-C
16	Op031No1-01		G:maj	06:23	05:44 (FG58)	07:19 (MC22)	435	E-E-D-R-C
17	Op031No2-01	Tempest	D:min	08:27	06:49 (FG58)	09:52 (MC22)	320	E-E-D-R-C
18	Op031No3-01	The Hunt	Eb:maj	08:29	07:53 (FG67)	09:07 (MB97)	341	E-E-D-R-C
19	Op049No1-01	Easy Sonata	G:min	04:35	03:41 (JJ90)	05:17 (MB97)	143	E-E-D-R-C
20	Op049No2-01	Easy Sonata	G:maj	04:37	04:19 (FJ62)	05:10 (MC22)	174	E-E-D-R
21	Op053-01	Waldstein	C:maj	10:38	09:25 (FG67)	11:36 (MC22)	387	E-E-D-R-C
22	Op054-01		F:maj	05:38	04:58 (AS35)	06:13 (MC22)	154	M1-Tr1-M2-Tr2-M3-C
23	Op057-01	Appassionata	F:min	09:35	07:35 (FG67)	10:39 (DB84)	262	E-D-R-C
24	Op078-01	A Thérèse	F#:maj	07:04	06:20 (FG58)	08:18 (MC22)	206	I-E-E-D-R-D-R
25	Op079-01	Cuckoo	G:maj	04:40	03:58 (AS35)	05:12 (MC22)	372	E-E-D-R-D-R-C
26	Op081a-01	Les adieux	Eb:maj	07:04	06:00 (FG67)	07:50 (DB84)	308	I-E-E-D-R-C
27	Op090-01		E:min	05:35	04:34 (FG67)	06:19 (MB97)	245	E-D-R-C
28	Op101-01		A:maj	04:00	03:35 (WK64)	04:29 (DB84)	102	E-D-R-C
29	Op106-01	Hammerklavier	Bb:maj	11:06	08:54 (AS35)	13:04 (DB84)	530	E-E-D-R-C
30	Op109-01		E:maj	03:46	03:14 (WK64)	04:19 (DB84)	99	E-D-R-C
31	Op110-01		Ab:maj	06:33	06:00 (FJ62)	07:33 (DB84)	116	E-D-R-C
32	Op111-01		C:min	09:05	08:20 (AS35)	10:04 (VA81)	209	I-E-E-D-R-C

Table 3.1: Overview of the first movements of Beethoven’s 32 Piano Sonatas. The table displays information including the work ID, trivial name (if applicable), global key, mean, minimum, and maximum duration of available recordings (see Table 3.3), number of measures, and the coarse structure. All durations are presented in the format mm:ss.

with the recordings. Examples include the MAPS [42], SMD [101], and Maestro datasets [64]. Hybrid acoustic–digital player pianos (with MIDI interfaces) or software synthesizers are utilized for these datasets to achieve precise alignments between symbolic and audio representations. The ASAP dataset is a comprehensive resource that comprises digital musical scores aligned with MIDI and audio performances [47]. Peter et al. [117] made further refinements to ASAP, incorporating note-wise alignments between scores and performances. MusicNet [136] is another important dataset comprising chamber music audio recordings (mostly involving the piano) with aligned MIDI information. Finally, we want to mention the Piano Concerto Dataset (PCD) [114], which features excerpts of separate piano and orchestra tracks, making it a valuable resource not only for AMT and but also for source separation.

While the BPSD includes piano music, it differs from the previously mentioned piano datasets in two significant ways. Firstly, it focuses on a specific corpus of musical relevance. Secondly, it includes sheet music in various machine-readable formats, along with several carefully aligned performances of the same musical works. This characteristic defines it as what we refer to as a *multi-version* dataset.

Similar multi-version datasets, based on other repertoires, have influenced the design and structure of the BPSD. One notable example is the Schubert Winterreise Dataset (SWD), a multimodal collection featuring various representations and annotations of Franz Schubert’s song cycle *Winterreise* [153]. The SWD comprises sheet music in different machine-readable formats, audio recordings of nine performances, and annotations such as musical keys, chords, or structural parts. This dataset enables a systematic study of Schubert’s musical language and style in *Winterreise*, also allowing for comparisons of annotations across different annotators and hierarchy levels. Similarly, the Wagner Ring Dataset (WRD) constitutes a multi-modal and multi-version resource focused on the extensive opera cycle *Der Ring des Nibelungen* by Richard Wagner [155]. It includes 16 recorded performances of the complete *Ring* cycle, with annotations aligned to a symbolic score representation based on a public-domain piano reduction. The dataset also comprises annotations on measures, key and time signatures, scenes, and singing voice regions.

In all three datasets (SWD, WRD, and our BPSD), a combination of manual work and semi-automated approaches has been employed for creating and transferring annotations between different versions, bridging the symbolic and audio domains. As detailed in Section 3.6, the multi-version property of datasets creates numerous research opportunities by utilizing the diverse information sources provided through sheet music and the various performances, reflecting a range of different perspectives on the same musical work. This not only facilitates the study of performance aspects [85] but also enables the investigation of algorithmic approaches for various tasks, including automatic music transcription [12], optical music recognition [23], music synchronization [103], cross-modal retrieval [102], chord recognition [116], structure analysis [111], and pattern discovery [95], just to name a few.

We conclude this section by discussing resources specifically related to our Beethoven scenario, focusing on the first movements of his piano sonatas. In [26], the authors introduce the BPS-FH dataset, which contains expert annotations on chords and harmonic functions for the sonata movements, referring to a symbolic encoding. These annotations include chord symbols and various interrelated chord functions, such as local keys and modulations, chord inversions, secondary chords, and chord quality. Continuing this work, the BPS-Motif dataset described in [69] offers note-level annotations of motives (characteristic parts of melodic themes) and their occurrences in the sonata movements. These annotations are well-suited for tasks such as motif retrieval and repeated pattern discovery. Finally, we want to mention the work presented in [71], which introduces automated methods for analyzing music recordings in sonata form. This approach is evaluated using audio recordings of the Beethoven corpus, incorporating structure annotations following Tovey’s analysis of the Beethoven sonatas [140]. Some of these annotations also serve as the foundation for the annotations provided by the BPSD, which have undergone careful revision, extension, and transfer to all versions.

Beyond the piano sonatas, Neuwirth et al. [109] released a symbolic dataset of Beethoven’s string quartets including all movements as symbolic scores together with harmony and phrase annotations. This dataset built the basis for a detailed statistical analysis of harmony [96]. Furthermore, there is a plenty of other corpora on piano music such as Mozart’s piano sonatas [66], as well as specific corpora on harmony [55, 67]

3.3. Data Organization and Primary Musical Material

Folder name	Content
- 0_RawData	Raw audio and symbolic data
- audio_ripped	Audio files as ripped from the CD
- WK64	
...	
- FG67	
- score_pdf_scan	Scanned score from IMSLP
- score_pdf_repetitions	Symbolic score in PDF format with repeat signs
- score_pdf_unfolded	Symbolic score in PDF format with unfolded repetitions
- score_sibelius_repetitions	Symbolic score in Sibelius format with repeat signs
- score_sibelius_unfolded	Symbolic score in Sibelius format with unfolded repetitions
- score_xml_repetitions	Symbolic score in MusicXML format with repeat signs
- score_xml_unfolded	Symbolic score in MusicXML format with unfolded repetitions
- score_midi	MIDI export of the symbolic score
- 1_Audio	Audio files with coherent structure
- 2_Annotations	Annotations with musical and physical timelines
- ann_score_note	Note events with start and end given in musical time
- ann_score_chord	Harmony annotations given in musical time
- ann_score_localkey	Local key annotations given in musical time
- ann_score_globalkey	Global key annotations
- ann_score_structureFine	Fine structure annotations given in musical time
- ann_score_structureCoarse	Coarse structure annotations given in musical time
- ann_audio_note	Note events with start and end given in physical time
- ann_audio_midi	Note events in physical time in MIDI format
- ann_audio_beat	Beat annotations given in physical time
- ann_audio_measure	Measure annotations given in physical time
- ann_audio_startEnd	Start and end of audio recordings (for removing silence/applause) given in physical time
- ann_audio_syncInfo	Alignment tuples for converting between musical and physical timeline
- ann_audio_modifications	Annotations for structural modifications of recordings
- ann_audio_chord	Harmony annotations given in physical time
- ann_audio_localkey	Local key annotations given in physical time
- ann_audio_structureFine	Fine structure annotations given in physical time
- ann_audio_structureCoarse	Coarse structure annotations given in physical time
- 3_Scripts	Python scripts to convert raw data into the structured format

Table 3.2: Overview of the folder structure of the BPSD. Score-based folders contain files named in the format `Beethoven_workID.ext`, while audio-based folders contain files in the format `Beethoven_workID_performerID.ext`.

or structure [53], which contain numerous piano works including Beethoven’s piano sonatas. All of these datasets focus on symbolic scores and expert-level annotations while not including audio recordings or multiple versions. Additional related work on datasets and applications is discussed in subsequent sections of this chapter.

3.3 Data Organization and Primary Musical Material

In this section, we provide an overview of the overall organization of the BPSD. In particular, we outline the global structure, introduce folder and file name conventions, discuss timeline conventions, describe the primary material (score and audio data), and summarize structural modifications for ensuring coherence.

3.3.1 Folder Structure

The BPSD is organized into four folders: `0_RawData`, `1_Audio`, `2_Annotations`, and `3_Scripts`. The folder `0_RawData` contains the raw audio files, directly extracted from the respective CDs without undergoing any structural modifications. Additionally, this folder includes sheet music representations in various formats. When necessary, editing operations are applied (see Section 3.3.6), and the modified audio files are stored in the folder `1_Audio`, where every version of a sonata adheres to a consistent structure. All annotations (see Section 3.5) are gathered in the folder `2_Annotations`. The subdirectories within this folder are named to indicate the type of annotation. Folders with the prefix `ann_score_` contain annotations with a musical timeline (given in measures) as specified by the score. The alignment of these annotations with individual audio versions results in annotations with a physical timeline (given in seconds), which are then stored in folders with the prefix `ann_audio_`. The directory `3_Scripts` contains all Python scripts used in creating the dataset. For a detailed folder structure overview of the BPSD, please refer to Table 3.2.

3.3.2 File Naming Convention

We encode score-based data using filenames in the format `Beethoven_workID.ext`, where `workID` indicates the work by opus and movement number, and `ext` signifies the type of annotation file. For example, the local key annotations based on a musical timeline for the first movement of Op. 53 (Waldstein Sonata) is stored at `2_Annotations/ann_score_localkey/` with the filename `Beethoven_Op053-01.csv`. For performance-related data, we append the performance ID to the filename. For example, an audio recording of the same sonata played by Wilhelm Kempff in 1964 is stored in `1_Audio/` with the filename `Beethoven_Op053-01_WK64.wav`.

3.3.3 Musical and Physical Timelines

The concept of “time” in music can generally be approached in two ways. There is the concept of *musical time*, specified by the composer in terms of measures and beats, which remains independent of the performance’s individual tempo shaping. In contrast, the events in a performed version of the piece can be defined in *physical time*, measured in seconds.

For the BPSD, we adopt the following convention for encoding time positions on the musical timeline. We utilize a decimal representation with the integer measure position before the decimal point and three digits after the decimal point to encode relative positions within a measure. As an example, in a 3/4 time signature, we encode the position of the start of the third beat in measure 137 as 137.666. Physical time is consistently represented in seconds with a precision of three decimal places.

3.3.4 Score Data

We offer score data in various formats, starting with a scanned version of the 1952 edition by Henle, which was chosen being in the public domain accessible at IMSLP (stored at `score_pdf_scan`). Using this version as a basis, we create sheet music for each movement using the music notation software “Sibelius” (`score_sibelius_repetitions`), adhering to the measure numbering of the original score. Additionally, we unfold the notated repetitions in Sibelius to achieve a continuous measure count (`score_sibelius_unfolded`). Both the original and unfolded versions are then exported as MusicXML and PDF files, stored at `score_xml_repetitions`, `score_xml_unfolded`, `score_pdf_repetitions`, and `score_pdf_unfolded`, respectively.

To enable playback of the symbolic scores in synthesizers, we export the unfolded sonatas in the MIDI format (`score_MIDI`). To eliminate the need for a MIDI or MusicXML parser and ensure the preservation of measure numbering, we also export all individual notes of a sonata in `.csv` files (in `ann_score_note`). These files contain the start and end of a note event specified on a musical timeline, the MIDI pitch number, the current time signature, and accents or playing styles like *staccato* or *sforzato*.

3.3.5 Audio Recordings

The BPSD comprises eleven complete performances of the first movements of Beethoven’s 32 piano sonatas. Four of these performances (AS35, FG58, FJ62, WK64) are in the public domain in the EU⁷ and are included with the dataset, while the remaining seven performances are commercially available. The recordings by Jank are available at IMSLP⁸, while all other recordings are uniquely identified using their European Article Number (EAN). We provide additional metadata such as MusicBrainz IDs, when available. An overview of all audio versions in the BPSD is presented in Table 3.3. To ensure consistency, we convert all audio files to mono WAV format with a sampling rate of 22 050 Hz.

3.3.6 Structural Modifications

In the original recordings forming the basis of the BPSD, we identified structural differences that resulted in notable inconsistencies in the musical timelines. These differences include the absence of repetitions in the exposition, additional repetitions of the development and recapitulation, and discrepancies in the number of measures played due to performance errors.

While concepts such as “Measure Maps” [54] have been introduced for transferring symbolic annotations between editions with different structures, we choose to apply structural modifications directly at the audio level. This approach ensures that audio and annotations are inherently coherent at every step of the

⁷ Please note that these recordings might not be in the Public Domain in other countries outside the EU and Switzerland.

⁸ https://imslp.org/wiki/Category:Jank,_Fritz

Chapter 3. Beethoven Piano Sonata Dataset: Alignment-Based Dataset Curation

ID	Performer	Year	Label	EAN Code	Orig. Dur.	Final Dur.
WK64	Wilhelm Kempff	1964	Deutsche Grammophon	028944796629	03:18:26	03:45:31
FJ62	Fritz Jank	1962	Instituto Piano Brasileiro	available at IMSLP	03:35:13	03:41:26
FG58	Friedrich Gulda	1958	Decca	028948514519	03:34:00	03:34:00
AS35	Artur Schnabel	1935	Warner Classics	0190295975050	03:31:03	03:33:35
MC22	Muriel Chemin	2022	Odradek	855317003615	04:08:22	04:05:11
MB97	Malcolm Bilson et al.	1997	Claves	7619931970721	03:52:23	03:46:08
AB96	Alfred Brendel	1996	Philips	028941257529	03:54:34	03:52:28
JJ90	Jeno Jando	1990	NAXOS	730099150224	03:41:06	03:39:14
DB84	Daniel Barenboim	1984	Deutsche Grammophon	028941375926, 028941376626	03:58:37	03:58:37
VA81	Vladimir Ashkenazy	1981	London Records	028944370621	03:48:16	03:46:27
FG67	Friedrich Gulda	1967	Amadeo	028947687610	03:25:02	03:25:02
Total					40:47:08	41:07:45

Table 3.3: Overview of audio recordings in the BPSD. The upper four performances with identifiers WK64, FG58, FJ62, and AS35 are in the public domain and freely accessible within the BPSD. All remaining recordings are commercially available and can be identified using the EAN code. Durations are presented in the format hh:mm:ss.

processing pipeline and can be immediately used in data-driven approaches without the need for ad-hoc modifications.

To this end, we designated the performances by Daniel Barenboim (DB84) as the reference version for all movements to ensure temporal coherence in the BPSD. It is noteworthy that Friedrich Gulda follows the same performance structure convention in both FG58 and FG67. Barenboim generally adheres to the notated score and closely follows the model often referred to as sonata form (“Sonatenhauptsatzform”), beginning with an exposition and its repetition, followed by a development, and concluding with a recapitulation. Only when explicit deviations from this form are notated in the sheet music, such as repeat signs for the recapitulation and development parts with alternative first and second endings and/or a closing coda, does Barenboim precisely follow the structure notated in the score. An overview of the reference structure for all sonatas underlying our dataset is provided in the last column of Table 3.1. For a more detailed discussion of the sonata form, see Section 3.5.3 and [68, 108].

Using DB84 as a reference, we edited the other performances to precisely follow the same structure. For this purpose, we applied three types of modifications: *cut* (removing a part completely), *copy* (duplicating a part and inserting it at a different time position), and *insert silence* (introducing silence in exceptional cases for short periods when a copy operation was not possible due to missing audio material for alternative endings). Annotations for these modifications are provided in separate CSV files for each recording that requires adjustment, available in `ann_audio_modifications`. All edit operations are automatically applied using a Python script. The finalized audio recordings are stored in the folder `1_Audio`.

As a result, the editing process guarantees coherence across all versions, adhering to the same musical timeline with a unique identifier for each measure, thus establishing well-defined alignments without gaps. Additionally, we ensured that the unfolded musical scores (Section 3.3.4) also adhere to the same

Table 3.4: BPSD: Absolute error in milliseconds for measure transfer heuristic. We report the mean, median, and 90th/95th percentiles of the error distribution.

Feature	Mean	Median	90th	95th
T1_SYM_O	66	17	115	234
T1_SYM_OF	52	20	102	160
T1_RAW_OF	41	12	70	121
T2_SYM_O	109	20	272	466
T2_SYM_OF	56	14	138	251
T2_RAW_OF	47	15	97	207
FB_RAW_OF	44	20	70	128

timeline conventions. Finally, we note that all annotations are exclusively specified for the modified music recordings and unfolded scores.⁹

3.4 Music Synchronization

Thanks to a coherent musical timeline across all versions of a movement, we can utilize music synchronization techniques to temporally align all score and audio representations. This approach enables us to initially specify annotations based on a shared musical timeline and subsequently transform this timeline to align with the physical timelines of specific performances. In this section, we outline the experiments on music synchronization techniques conducted during the creation of the BPSD. For a detailed description of the transcription and synchronization approach we refer to [169] and Section 2.6.

Notation

We briefly recall from Section 2.6.1 the terminology used in the following. We consider three feature extraction approaches: a classical filterbank (FB) and two transcription models (T1 [90], T2 [77]). Features are represented either as continuous pianoroll-like probabilities (RAW) or as discretized binary symbolic features (SYM). For alignment, we use two DTW-based variants: an onset-only approach using standard DTW (O) and a combined onset–frame approach with MrMsDTW (OF) [103], see also Table 2.1 for a summary.

3.4.1 General Synchronization Performance

We now analyze the overall matching of score–audio and audio–audio synchronization on the BPSD by using the measure-transfer heuristic with reference measure annotations for the WK64 recordings as

⁹ We are aware that such modifications are unusual from an editor’s perspective or for performance analysis. However, regarding the BPSD mainly as a resource for studying algorithmic approaches to MIR and computational musicology, we value the structural consistency across versions as favorable. Please note that original audio files are still accessible in the folder `0_RawData` and the scripts provided along with the data ensure reproducibility and bridge the link back to the original data

described in Section 2.6.3 and illustrated in Figure 2.10b. In Table 3.4, we report the mean, median, 90th and 95th percentiles for the absolute error between measure positions obtained from score–audio and audio–audio transfer.

Analyzing the median absolute error in Table 3.4, all features yield a precision between approximately 12 ms and 20 ms, without a clear tendency towards one particular method. However, it is the robustness (measured by the 90th and 95th percentiles of the error distribution) where we find a clear trend: using only onsets from symbolic features (SYM_O) yields large alignment outliers, with the 90th percentile of the absolute error being 234 ms for T1 and even 466 ms for T2. Using additional frame features (SYM_OF) lowers the 95th percentile to 160 ms and 251 ms, respectively. In line with our observations on the ASAP dataset (see Section 2.6), using the raw predicted likelihoods (RAW_OF) further reduces the mean as well as the percentiles for both transcription models. The T1 transcriber, which was trained on audio from the acoustically diverse MusicNet [136] dataset, exhibits significantly lower errors than T2 (121 ms vs. 207 ms for the 95th percentile), which was trained only on MAESTRO. While using filterbank features (FB) resulted in relatively high errors on ASAP (see Section 2.6), on the BPSD we observe metrics that are similar to those of the T1 transcriber, and considerably better than those of the T2 model. This indicates a lack of robustness of the DL-based transcription models on the diverse acoustic conditions of the BPSD, as we would expect DL methods to perform better than simple filterbank approaches. In the following section, we will investigate the cause of this problem and explore strategies for mitigation.

3.4.2 Detailed Analysis and Finetuning

To further investigate why the alignment pipelines using transcription features (T1, T2) do not yield significantly better results on the BPSD than those features using the filterbank baseline (FB), we further break down the investigation to the BPSD’s individual audio versions. Transcription models (and DL systems in general) are known to exhibit a degraded performance when there is a domain shift between the test data and the training data [41, 121]. Such effects can be caused by poor audio quality in general, or, for music data, by a difference in timbre or tuning.

Identifying problematic versions. We restrict our analysis to raw frame and onset features (RAW_OF) from the filterbank (FB) and the T1 transcriber, as these showed the overall most robust results on the complete BPSD (see Section 3.4.1) and illustrate the median and 95th percentile of the error distribution for all audio versions of the BPSD in Table 3.5. Analyzing the 95th percentiles for T1_RAW_OF in Table 3.5, we identify two problematic versions, namely the 1958 recordings by Friedrich Gulda (FG58) with 788 ms and the 1997 recordings by Malcolm Bilson et al. (MB97) with 146 ms. By inspection of the recorded pieces, we find two different reasons for the alignment instabilities.

Musical and acoustic reasons for instabilities. Friedrich Gulda recorded his first cycle of Beethoven’s Piano Sonatas (FG58) over a relatively long time span between 1950 and 1958, playing different pianos in different environments. Among the FG58 recordings, we identify Sonata No. 26 (“Les adieux”) and No.

Table 3.5: Median and 95th percentile of the absolute synchronization error for individual performances in the BPSD. All experiments use raw frame and onset features (RAW_OF). Values are given in milliseconds.

Version	FB_RAW_OF		T1_RAW_OF		T3_RAW_OF	
	med.	cf 95.	med.	cf. 95	med.	cf. 95
AB96	19	132	11	58	12	58
AS35	20	157	11	62	12	76
DB84	20	149	12	71	14	93
FG58	19	137	27	788	12	80
FG67	20	138	10	49	11	59
FJ62	22	185	11	80	13	84
JJ90	17	102	10	56	12	56
MB97	23	217	12	146	12	103
MC22	22	178	13	80	14	89
VA81	19	143	11	61	12	69
WK64	16	99	10	49	11	45
average	20	128	12	121	12	62

29 (“Hammerklavier”) as especially problematic, showing large differences in tuning, along with high background noise.

The pianist Malcolm Bilson (MB97) is committed to historically informed performance practice. His interpretations on historical instruments introduce a novel approach to performance in an era predominantly defined by the use of modern instruments. Malcolm Bilson and colleagues recorded their 1997 cycle of Beethoven’s Piano Sonatas on nine fortepianos, including original historical instruments. Compared to modern pianos, the overall sound of fortepianos is significantly different, due to different mechanics, strings, and resonance bodies. Furthermore, the timbre varies across registers, e.g., bass notes sound fundamentally different compared to high-octave notes, and the reference pitch deviates from today’s standard of 440 Hz (A4). In summary, these deviations in timbre, tuning, and recording noise can be regarded as a so-called “domain shift”, i.e., the FG58 and MB97 recordings are not close enough to the transcriber’s training data. As a result, the model’s predictions are highly unstable and often do not correspond to the actual played notes.

Fine-tuning the transcriber. Our goal is to obtain highly accurate alignments on the BPSD, despite the aforementioned issues. Therefore, we choose to adapt the T1 transcriber to the BPSD’s audio versions by fine-tuning the model on the target data itself, using an EM-based training strategy described in [90]. Note that this is a valid procedure for the purpose of this study, as our goal is not to assess transcription performance, but rather to align the audio with score, which is known at test time (see [90] for a detailed description of the training process using unaligned pairs of audio and score data). Therefore, we do not overfit the model towards a reference alignment. We denote the fine-tuned transcriber as T3.

Results with fine-tuned transcriber. After fine-tuning on the BPSD, the T3 model significantly improves the 95th percentile for the two problematic versions FG58 and MB97 from 788 ms to 80 ms and from 146 ms to 103 ms, respectively. For all other audio versions, the median absolute error and the 95th percentile of the fine-tuned transcriber T3 remain in a similar range as the original model T1. We note that the

averaged 95th percentile of 62 ms for the fine-tuned transcriber T3 is in the range of the typical tolerance in beat-tracking applications (70 ms), making the proposed synchronization approach with raw features useful for the creating datasets with high demands regarding temporal precision.

3.4.3 Discussion of Synchronization Accuracy

Recall that the measure positions for WK64 were annotated manually (see also Section 3.5.1). Thus, the absolute errors reported for WK64 provide an indicator of the performance of audio–score synchronization alone (along with inaccuracies in the manual annotation process that, for more complex music, can reach a level of up to 100ms [149]). While yielding slightly better results, the errors on WK64 remain in the same order of magnitude as the other results.

Overall, we may conclude that the synchronization yields accurate alignment results with median errors below 15 ms (assuming that it behaves in a similar fashion within measure positions). The 95th percentile of the error distribution of 62 ms on average provides a more generous estimate of the accuracy, which needs to be taken into account when working with automatically transferred annotations (see Section 3.5).

3.4.4 Aligning Musical and Physical Timelines

Using the music synchronization method described before, we now describe the process of aligning the musical and physical timelines across all performances. In the context of Western classical music, audio–audio synchronization often proves to be more robust than score–audio synchronization. This is attributed to the fact that input features are computed from data within the same domain in the audio–audio case.

Motivated by this and following [153, 155], we adopt a multi-step approach to align the musical and physical timelines. In the initial step, audio–audio synchronization is applied to transfer *measure annotations* from the WK64 recordings to all other recordings. To mitigate boundary artifacts resulting from silence or applause at the beginning or end of a recording, we manually annotated the start and end points of each recording (`ann_audio_startEnd`). These time positions serve as anchor points in the synchronization approach (see [120] for an explanation of the anchor point concept).

In the second step, using the transferred measure positions as anchor points, we apply score–audio synchronization. This maps all *note onsets and offsets* from the musical timeline to the physical timeline, providing a representation of all symbolic note events precisely aligned with an audio version (available in `ann_audio_note` as a list of note events and in `ann_audio_midi` as a playable MIDI file for each track).

In the third step, the aligned note events serve as the basis for mapping *musical time to physical time*. Recall from Section 3.3.3 that each note onset of the score is encoded by a real-valued position on a continuous measure axis, while the aligned position in an audio version is a real-valued number encoding physical time

in seconds. This information is stored as tuples “(musical time, physical time)” in `ann_audio_syncInfo` and is utilized for every transfer from a score-based version to an audio-based version or vice versa. We use linear interpolation between note events to obtain a continuous mapping between positions on the musical timeline and a physical timeline.

It is crucial to emphasize that our approach transfers note information from the score, aligning the musical timeline of the score with the physical timeline of the performance. Consequently, the annotations are not tailored to individual note-level deviations within each performance, especially playing errors. In particular, our synchronization approach is not designed to handle arpeggios or trills. Hence, our note annotations should be regarded not as a direct transcription of the audio recording but as a temporal adaptation of the score information.

3.5 Annotations

In this section, we provide an overview of the annotations included in the BPSD. These annotations cover measure and beat positions, harmonic segments (local key and chord), and structural elements (on a coarse and fine level). For each movement, the annotations are available with respect to the musical timeline (for the score version) and with respect to performance-specific physical timelines (for all audio versions). For an overview, we refer to Figure 3.2.

3.5.1 Measure and Beat Annotations

As already mentioned in Section 3.4, we manually annotated measure positions for WK64, which serves as our reference version for measure annotations. The reasons for selecting WK64 as a reference include its availability in the public domain, its high audio quality, and Kempff’s choice of a relatively slow tempo, which simplifies the manual annotation process. In this process, a musically trained listener annotated all WK64 recordings using the Sonic Visualiser software [24]. To ensure high quality, these annotations were processed with a sonification tool and cross-checked by an independent and experienced audio annotator. The experiments in Section 3.4 suggest that the annotation accuracy is within the order of or better than the synchronization accuracy of 20 to 30 ms, which is better than obtained for complex orchestral music in [48, 149]. The measure annotations for the other audio versions were then obtained using audio–audio synchronization techniques and stored in `ann_audio_measure`.

To derive beat annotations for the score, we initially generated a list of all beats in a movement by dividing each measure by the number of beats specified in the time signature. Subsequently, utilizing the alignment of musical and physical timelines (see Section 3.4.4), we transferred the beat positions to the physical timelines of the audio versions, storing them in `ann_audio_beat`.

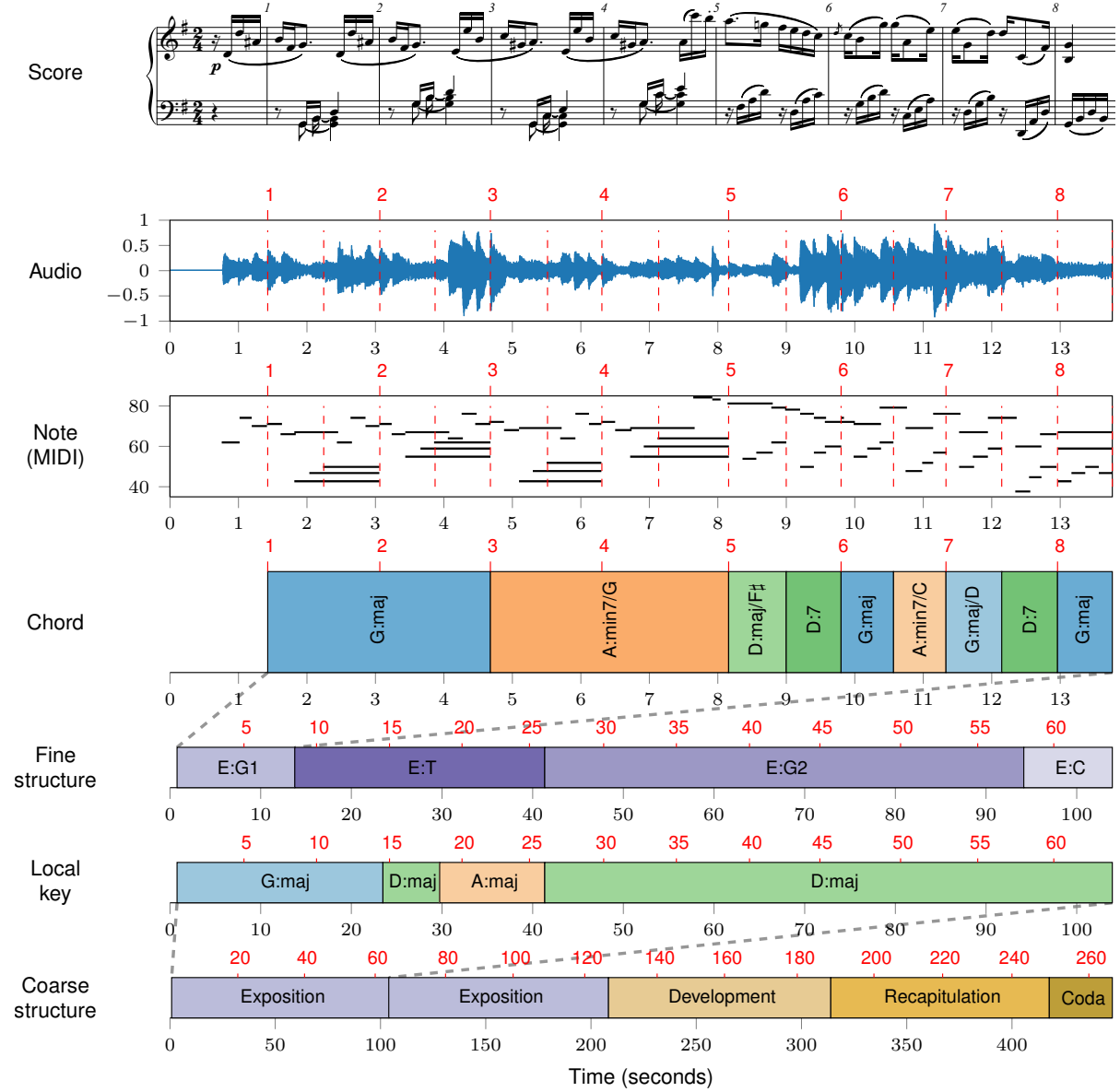


Figure 3.2: Overview of various annotations in the BPSD illustrated using the first measures of the Sonata Op. 14, No. 2 in G Major. Measure positions are marked with red ticks, while beat positions are indicated by red dashed lines.

3.5.2 Harmony Annotations

The BPSD includes diverse harmony-related annotations in the form of absolute and relative chord, local key, and global key annotations. For the chord annotations, we started with the score-based annotations provided by the BPS-FH dataset [26]. We adjusted the original timelines, initially specified on a quarter note grid, to our measure-based musical timelines, represented by real-valued measure counts. Additionally, we made structural modifications to ensure coherence with the reference musical timeline when necessary. Finally, these annotation were transferred to the audio versions using the alignment information. As a

first type of chord annotations, we include the high-level chord labels from the original BPS-FH dataset consisting of chord functions (as scale degrees in Roman Numeral notation) relative to the local key. To facilitate the usage of BPSD for tasks like audio-based chord estimation, we additionally opt for a more pragmatic encoding.¹⁰ Therefore, we choose to provide absolute chord information (root notes specified as pitch classes) in the encoding scheme proposed by Harte et al. [62]. Enhancing the applicability of chord annotations for various tasks, we represent chords at different levels of detail. This includes shorthand notation (e.g., C:7/E), extended form (e.g., C:(3,5,b7)/E), major/minor with inversion (e.g., C:maj/E), major/minor (e.g., C:maj), and a numerical identifier of the latter (e.g., 1). Chord annotations with the start and end of each chord in musical and physical time are provided in `ann_score_chord` and `ann_audio_chord`, respectively.

In a similar manner, we include the local key annotations from the BPS-FH dataset [26], adapting and transferring them accordingly for the audio representations. The local key annotations, along with their start and end times in both musical and physical timelines, are accessible in `ann_score_localkey` and `ann_audio_localkey`, respectively.

Finally, for the global key annotations, we deviate from our convention of having one annotation file per movement and performance. Instead, we provide a single table in `ann_score_globalkey`, containing the global keys of the first movements of all 32 piano sonatas.

3.5.3 Structure Annotations

As a last type of annotation, we include structural information, primarily intended for the purposes of overview and navigation. Strictly following the analysis by Tovey [140], we include structural boundaries in two levels of granularity. In particular, we provide boundary annotations for the coarse structure (such as Exposition, Development, Recapitulation, etc.) and fine sub-structures based on thematic material used (such as First Group, Second Group, Transition phases, etc.), as visualized in Figure 3.2 and described in the following section.

The sonata form is a prominent structural concept in Western classical music and can be categorized into five types [68]. In Beethoven’s piano sonatas, the type 3 sonata (“Sonatenhauptsatzform”) is predominantly used, usually containing a (repeated) exposition, a development, and a recapitulation part, occasionally starting with an introduction and concluding with a coda. Following the analysis by Tovey [140], we provide this coarse structural information in relation to the musical timelines in `ann_score_structureCoarse` and the performance-specific physical timelines in `ann_audio_structureCoarse`.

In addition to the overarching sonata form structure, one often encounters specific substructures within the exposition and recapitulation. Typically, the exposition introduces the primary thematic material of the movement through two tonally contrasting subject groups. The first subject group is usually in the global

¹⁰ See, for instance, the mir-eval library (https://github.com/craffel/mir_eval).

key (home key) of the movement, while the second subject group is in the dominant key (for sonatas in major) or the relative key (for sonatas in minor). These subject groups are commonly linked by a modulating transition, and the exposition frequently concludes with an additional closing theme or codetta. The recapitulation mirrors the subparts of the exposition but incorporates a significantly different local key progression. Also, it often includes prolonged sections, introduces new material, and incorporates local modulations.

Consistent with the analysis by Tovey [140], we offer structure annotations on a more detailed level in `ann_score_structureFine` for the score and in `ann_audio_structureFine` for the audio recordings. Comparable annotations were employed in the study by Jiang and Müller [71] to assess a computational approach for analyzing music recordings in sonata form.

While we are aware of the limitations of Tovey’s analysis, assuming a rather outdated view on sonata form, the great level of detail and descriptiveness (“bar-by-bar”) is an advantageous property in our case since it allows for deriving clear marks of reference that facilitate navigation and overview.

We suggest a careful interpretation of structure annotations, as they often capture tendencies rather than strict rules. In practice, numerous exceptions and modifications to the sonata form exist, and the utility of the sonata form as an oversimplifying model is a subject of debate among many researchers, as discussed in [76, 108].

3.6 Application Scenarios

In this section, we outline several use cases to highlight the research potential of the BPSD in fields such as MIR and computational musicology. Some of these studies draw upon prior research that utilized similar datasets.

3.6.1 Learning Pitch-Class Representations

Pitch-class or chroma representations extracted from audio recordings play a fundamental role in various MIR tasks. Traditional chroma features, computed using signal processing methods, are often impaired by timbral properties like overtones or vibrato, resulting in an only rough correspondence to the pitch classes indicated by a musical score. To overcome this, [154] employed a DL approach for learning transcription-like pitch-class representations using synchronized score–audio pairs from classical music. Providing aligned training pairs, the BPSD proves to be a valuable resource for training such DL-based models. An example of an audio-based pitch-class representation learned with a CNN-based model and trained exclusively on the BPSD is illustrated in Figure 3.3.

As another central feature of BPSD, its coherent structure enables the exploration of different systematic training–test splits, including the cross-work split (across movements) and the cross-version split (across

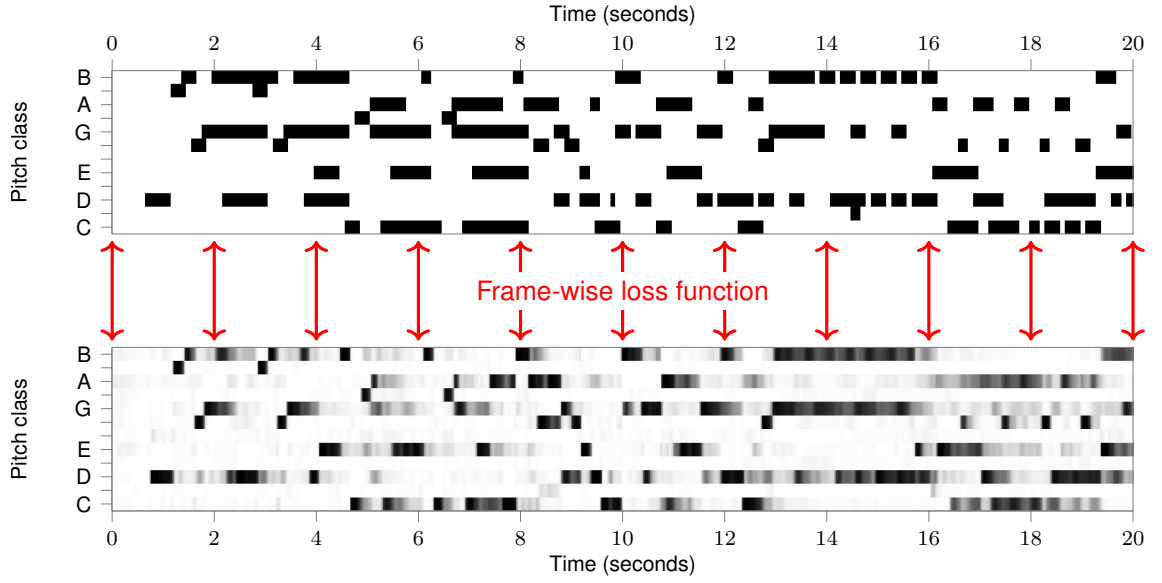


Figure 3.3: Synchronized score–audio training pair for learning pitch-class representations using a frame-wise loss function.

performances). Introduced in [152], such an approach was applied to study the generalization properties of data-driven methods for local key estimation using the Schubert Winterreise Dataset [153].

3.6.2 Local Key Analysis

The aforementioned study leads us directly to our next use case, where we consider the task of estimating local keys—a task that can be seen as conducting harmony analysis on a mid-level time scale. Instead of explicitly extracting and numerically evaluating such information, it is often insightful to visualize harmonic structures and leave the final interpretation to a music expert. One such approach is described and applied to Beethoven’s sonatas in [151], where relevant pitch content with respect to the twelve diatonic scales is extracted from an audio recording and visualized in the form of a time–diatonic representation.

Figure 3.4 shows such a visualization of a time–diatonic representation derived from a WK64 recording of the Piano Sonata Op. 14 No 2 in G Major. Note that the visualization is organized along the recording’s physical timeline. The reference annotations for local keys, indicated by overlaid red rectangles, assist in evaluating the estimation results. The automatically computed time–diatonic representation offers a clear overview of the general harmonic progression, emphasizing the typical key structure of the sonata form.

The figure also demonstrates that the inclusion of additional information, such as the start times of musical form sections or the position of measures, can further enrich the visualization. Such annotations are particularly helpful when using audio-based visualizations in combination with musical scores for scrutinizing and gaining a deeper understanding of the phenomena revealed by the visualizations.

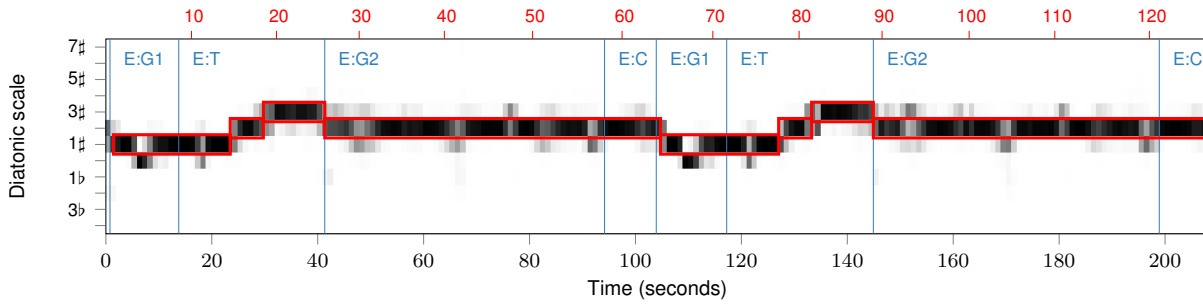


Figure 3.4: Visualization of a time–diatonic representation derived from the WK64 recording of the first movement of the Piano Sonata Op. 14 No 2 in G Major. The local-key reference annotations are indicated by the overlaid red rectangles.

3.6.3 Chord Recognition

In our third use case, we consider the task of chord recognition, which can be seen as a detailed analysis of harmony [116]. In the early cross-version study by Konz et al. [78], the authors showed that chord recognition results can be stabilized by simultaneously performing this task for several audio recordings and then merging the results using a fusion approach (e.g., majority voting).

The BPSD proves ideal for such a cross-version study, encompassing scores and multiple recordings for all movements. Figure 3.5 illustrates this methodology, showcasing chord recognition results for all eleven performances of the initial measures of the Piano Sonata Op. 31 No. 1 in G major, alongside the majority vote and reference annotations. In this example, leveraging the improved pitch-class representations outlined in Section 3.6.1, we employed a simple template-based chord recognition approach, simplifying to the 24 major and minor triads. Furthermore, we transformed the performance-dependent physical timelines of the chord recognition results into a unified musical timeline based on alignment annotations. Finally, we consolidated the results via majority voting, resulting in increased stability and improvement compared to individual recognition outcomes.

3.7 Further Notes

This chapter presented the carefully curated and annotated multi-version BPSD, centered around the first movements of Beethoven’s piano sonatas. We provided a comprehensive overview of the data, addressing our methodologies for curation, processing, correction, and annotation. The dataset includes symbolic scores in various formats and over 40 hours of audio data from eleven performances, with four being in the public domain accessible for research purposes. Secondary materials encompass alignments, measure and beat positions, chord and local keys, and structure annotations for all versions.

We estimate that the creation of the BPSD required more than 2000 hours in total, involving tasks such as symbolic score creation (ca. 500 hours), annotation creation and adjustment (ca. 500 hours), programming

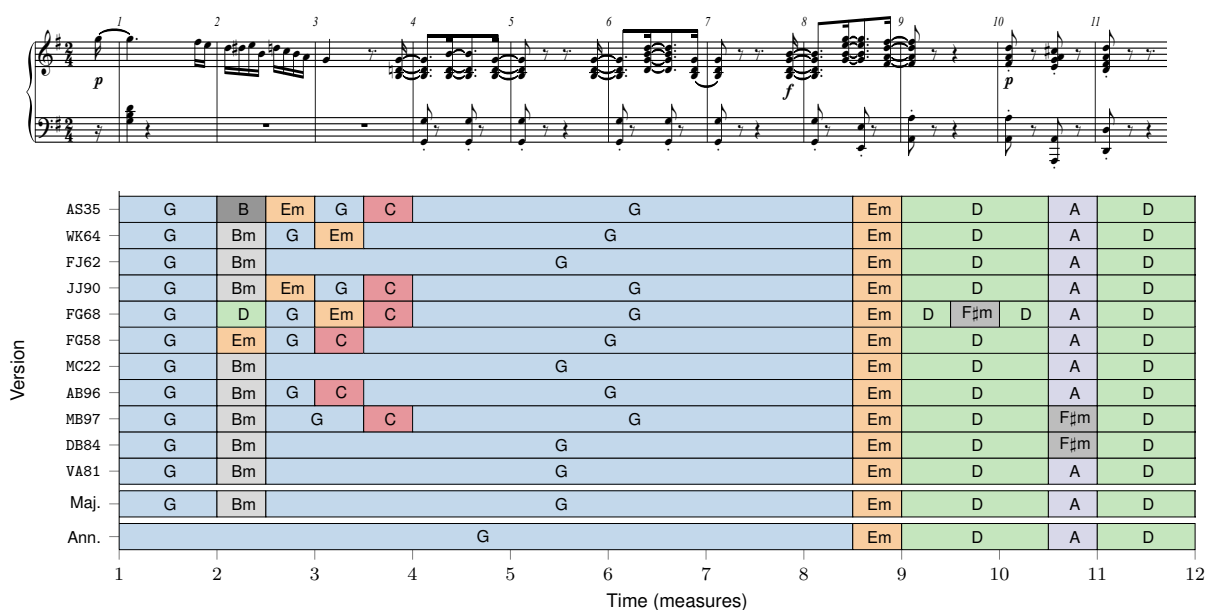


Figure 3.5: Cross-version chord recognition for the initial measures of the Piano Sonata Op. 31 No. 1 in G major. The results are presented for all eleven performances, alongside the majority vote and the reference annotations.

(ca. 500 hours), and further processing and refinement (ca. 500 hours), excluding the actual research work with the data.

A key feature is the dataset’s coherence, achieved by enforcing unified musical time for all versions and ensuring annotation consistency. Leveraging this coherency (admittedly sometimes achieved through simplification and musicologically debatable assumptions), the BPSD is designed to be technically convenient and can be readily utilized for training and testing machine learning approaches. This includes the investigation of data-driven approaches for chord recognition, local key estimation, measure and beat tracking, structure analysis, and music transcription to evaluate their capabilities for generalization across works, versions, and modalities.

Building upon previous studies, we highlighted various research opportunities provided by the dataset. Specifically, we hope that the BPSD will foster further engagement in the musicological discourse on Beethoven’s piano sonatas and their performances by offering a framework that allows for the easy transfer of musicological annotations, typically defined on a musical timeline, to a multitude of audio recordings. In this context, we believe that the development of interactive software and visualization applications holds significant potential for interdisciplinary research and educational applications. In conclusion, we aspire for the BPSD to be a valuable resource and a source of inspiration, offering compelling research opportunities in the fields of music information retrieval, computational musicology, and digital humanities at large.

4 Background: Soft Dynamic Time Warping (SDTW)

In the previous chapters, alignment was treated as a stand-alone algorithmic component that produces a strong alignment between two time sequences with frame-level correspondence. Starting from weakly aligned data, this procedure converts different representations of the same musical piece into a common timeline that can then be used for analysis or supervised learning.

In many machine learning scenarios, however, treating alignment as a separate preprocessing step is undesirable, since it reduces the alignment to a single hard path and discards information about the probability of alternative alignments. Instead, we aim to train models directly from weakly aligned data by integrating the alignment process into the optimization objective. Classical DTW, however, is defined as the hard minimum over alignment path costs and therefore does not provide useful gradients for end-to-end optimization.

To address this limitation, this chapter introduces Soft Dynamic Time Warping (SDTW), a differentiable relaxation of DTW. We derive its forward and backward computations, establish the notation used throughout the subsequent technical chapters, and analyze how the SDTW parameterization influences the resulting loss landscape and gradient behavior. These results provide the mathematical foundation for the differentiable alignment methods developed in the remainder of this thesis.

4.1 Related Work: Deep Learning and Differentiable Alignment

DNNs have become central to many tasks involving sequential data, including ASR [58], motion alignment [25], and several applications in MIR [12]. In the latter, DNNs are widely used for music transcription, PCE [80, 154], chord recognition [81], and lyrics alignment [135]. Learned representations such as deep chroma features [80, 154] have become standard inputs for downstream tasks including synchronization [113], audio thumbnailing [9], and cross-version analysis [128].

A major challenge in training such models is the availability of accurately aligned input–target pairs. Strongly aligned targets, i.e., frame-level annotations, typically require costly manual labeling. In music transcription, for example, this involves annotating note events with precise onset and offset times. In contrast, weakly aligned targets are often easier to obtain. These may consist of globally corresponding

input–target pairs where the correct order of events is known, but precise temporal alignment is missing—for instance, note sequences derived from a score that correspond to an audio segment without specifying exact time positions. Training with such weak supervision requires incorporating an alignment step before or during the loss computation.

The first strategy is to decouple alignment and learning. Several works perform offline DTW alignment between predictions and weak targets, treat the aligned outputs as strong targets, and iteratively refine the alignment in an EM-like procedure [90, 121]. While effective in practice, such approaches require repeated offline alignment steps and do not provide a unified differentiable objective.

A second approach for weakly supervised sequence learning that integrates alignment into the loss computation is the CTC loss [59]. Originally developed for ASR, CTC aligns network predictions with sequences of discrete labels by marginalizing over monotonic alignment paths. It has been successfully applied in lyrics alignment [135] and musical theme retrieval [160], and has been extended to multi-label problems such as Multi-Pitch Estimation (MPE) [147]. However, CTC assumes a discrete label alphabet and becomes algorithmically complex in high-dimensional multi-label settings. Moreover, it does not naturally support continuous-valued targets.

An alternative alignment paradigm is DTW [98]. To enable gradient-based DNN optimization in the presence of weakly aligned input–target pairs, differentiable variants of DTW have been proposed by replacing the minimum operator in the DP recursion (see Section 2.4) with smooth approximations [30, 61, 94]. The most prominent formulation, SDTW [30], replaces the hard minimum operator with the softmin function. SDTW has been applied to performance–score synchronization [1] and classification tasks such as MPE [83]. While improved mathematical formulations have been proposed [17], SDTW-based training can exhibit instabilities, particularly when input and target sequences differ substantially in length [83].

4.2 Differentiable Approximation of the Minimum Function: Softmin

The classical DTW algorithm, as described in Chapter 2, employs the hard minimum function, which does not have a continuous derivative. To this end, we replace the hard minimum function with a differentiable approximation, the so-called softmin function. We refer to Appendix B for more details about the derivation of the softmin function. Note that, while Appendix B defines softmin for vectors in $\mathbb{R}^D$, in the following we will use the softmin function with inputs of variable length. Therefore, let I be a finite index set, let $Q = (q_d)_{d \in I}$ with $q_d \in \mathbb{R}$ be a collection of real numbers indexed by I , and let $\mathbb{R}^I = \{Q : I \rightarrow \mathbb{R}\}$ denote the set of real-valued functions on I . To this end, the softmin function $\mu_{\text{soft}}^\gamma : \mathbb{R}^I \rightarrow \mathbb{R}$ is defined as

$$\mu_{\text{soft}}^\gamma(Q) := -\gamma \log \sum_{d \in I} \exp(-q_d/\gamma) , \quad (4.1)$$

4.2. Differentiable Approximation of the Minimum Function: Softmin

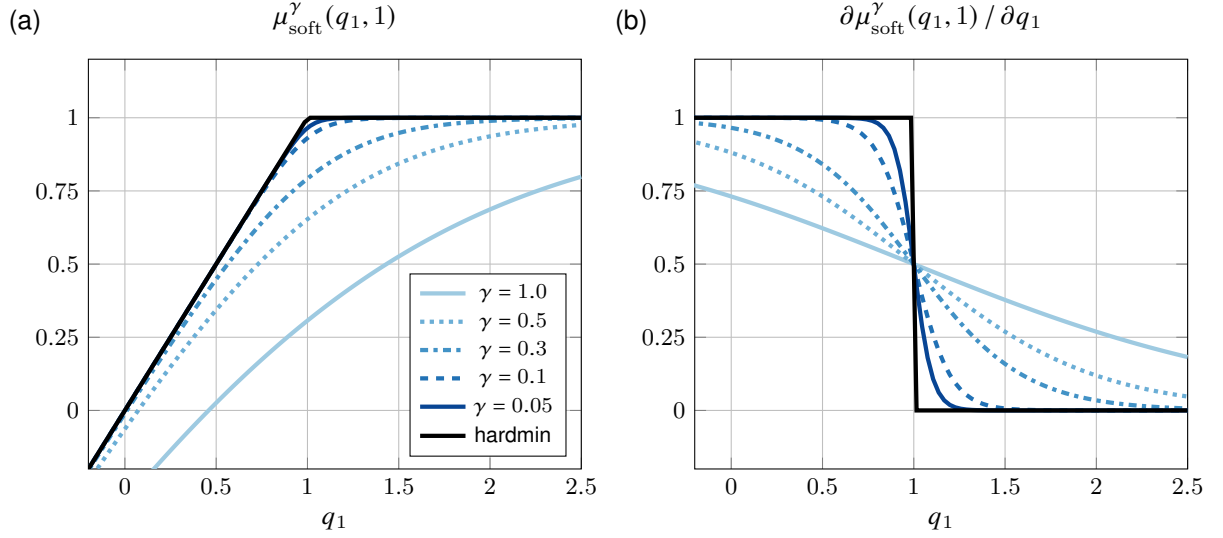


Figure 4.1: Illustration of (a) the softmin function and (b) its derivative for different temperature parameters γ .

where $\gamma \in \mathbb{R}_{>0}$ denotes a temperature parameter [30]. The partial derivatives of the softmin function form a *softargmin* distribution over the indices:

$$\frac{\partial \mu_{\text{soft}}^\gamma(Q)}{\partial q_d} = \frac{e^{-q_d/\gamma}}{\sum_{i \in I} e^{-q_i/\gamma}}, \quad (4.2)$$

sometimes referred to as *Boltzmann* or *Gibbs* distribution. An illustration of the softmin function and its derivative for different temperature parameters is provided in Figure 4.1. For brevity, when the softmin function is applied to a finite collection of real numbers (a, b, c) , we write $\mu_{\text{soft}}^\gamma(a, b, c)$ instead of $\mu_{\text{soft}}^\gamma((a, b, c))$. For example, in Figure 4.1 we visualize the softmin function for a tuple $(q_1, 1)$, where we vary q_1 .

The temperature parameter γ of the softmin function can be used to control the smoothness of the approximation of the softmin function. As shown by Theorem B.2, for $\gamma \rightarrow 0$, the softmin function approximates the hard minimum function:

$$\lim_{\gamma \rightarrow 0} \mu_{\text{soft}}^\gamma(Q) = \min_{d \in I} q_d. \quad (4.3)$$

In this limit case, the gradient of the softmin function yields a uniform distribution over the minimizing indices $I^* := \{i \in I \mid q_i = \min_{d \in I} q_d\}$:

$$\lim_{\gamma \rightarrow 0} \frac{\partial \mu_{\text{soft}}^\gamma(Q)}{\partial q_d} = \begin{cases} \frac{1}{|I^*|}, & \text{if } d \in I^*, \\ 0, & \text{else,} \end{cases} \quad (4.4)$$

which becomes a one-hot vector if there is a unique minimum in Q , see Theorem B.3. In contrast, when γ takes large values, the softmin function transitions from a minimum-like operator to a mean-like function,

as given by Theorem B.2:

$$\mu_{\text{soft}}^{\gamma}(Q) \xrightarrow{\gamma \rightarrow \infty} -\gamma \log D + \frac{1}{D} \sum_{d \in I} q_d, \quad (4.5)$$

where $D = |I|$ is the cardinality of the inputs, and the gradient can be approximated by a uniform distribution (Theorem B.3):

$$\lim_{\gamma \rightarrow \infty} \frac{\partial \mu_{\text{soft}}^{\gamma}(Q)}{\partial q_d} = \frac{1}{D} \quad \text{for all } d \in I. \quad (4.6)$$

More details about the softmin function, including an analytical derivation via convex optimization, are provided in Appendices A and B.

4.3 SDTW: Global Formulation

We now illustrate how the SDTW formulation originates as a differentiable relaxation of classical DTW, closely following [30]. We begin by writing out DTW as a minimum-cost path finding problem over a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$, as presented in Section 2.4. Alignments are denoted by binary alignment matrices $\mathbf{A} \in \{0, 1\}^{N \times M}$, and the set of all valid alignment matrices for sequences of length N and M is denoted by $\mathcal{A}$.

4.3.1 SDTW Cost

Recall from Section 2.4 that the DTW cost function $\text{DTW} : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}$ is given by computing the minimum cost over all possible alignments:

$$\text{DTW}(\mathbf{C}) = \min_{\mathbf{A} \in \mathcal{A}} (\langle \mathbf{A}, \mathbf{C} \rangle). \quad (4.7)$$

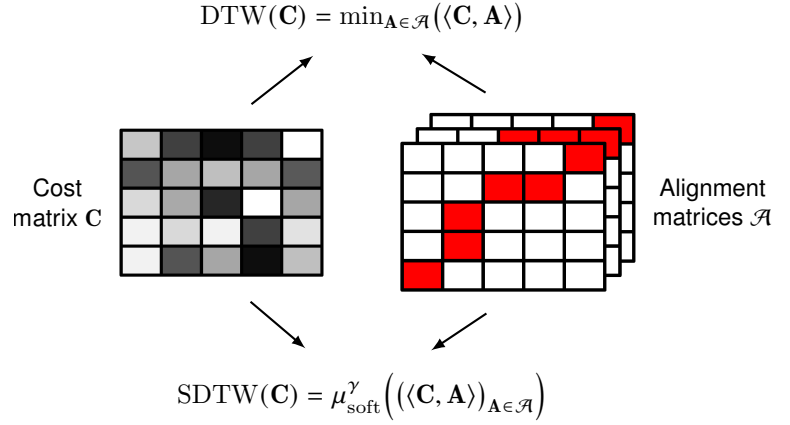
While the computation of the path cost $\langle \mathbf{A}, \mathbf{C} \rangle$ in (4.7) is a linear function and thus has a continuous derivative over $\mathbf{C}$, the minimum function itself does not. Therefore, we replace the minimum function by the softmin function $\mu_{\text{soft}}^{\gamma} : \mathbb{R}^I \rightarrow \mathbb{R}$, as defined in Section 4.2, and define the SDTW cost $\text{SDTW} : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}$ as in [30] by:

$$\text{SDTW}(\mathbf{C}) = \mu_{\text{soft}}^{\gamma} (\langle \mathbf{A}, \mathbf{C} \rangle_{\mathbf{A} \in \mathcal{A}}) \quad (4.8)$$

$$= -\gamma \log \sum_{\mathbf{A} \in \mathcal{A}} \exp (-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma). \quad (4.9)$$

An illustration of the DTW and SDTW formulations is given in Figure 4.2.

Figure 4.2: Illustration of DTW and SDTW cost computation. While DTW returns the cost along a single minimizing path, SDTW computes a soft minimum over all path costs.



4.3.2 SDTW Gradient

While the gradient of DTW w.r.t. the local cost matrix $\mathbf{C}$ yields a single cost-minimizing alignment path (see (2.18)), the SDTW gradient yields a probability distribution over paths (see Figure 4.3). In the following, we outline the computation of these “soft alignments”.

For calculating the gradient of the SDTW cost function w.r.t. the local cost matrix $\mathbf{C}$, we first write it in compact form

$$\text{SDTW}(\mathbf{C}) = \mu_{\text{soft}}^{\gamma} ((\langle \mathbf{A}, \mathbf{C} \rangle)_{\mathbf{A} \in \mathcal{A}}) \quad (4.10)$$

$$= -\gamma \log Z(\mathbf{C}) \quad (4.11)$$

using the partition function

$$Z(\mathbf{C}) := \sum_{\mathbf{A} \in \mathcal{A}} e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma}. \quad (4.12)$$

Taking the derivative of the SDTW function w.r.t. the elements $\mathbf{C}(n, m)$ of the cost matrix, we obtain:

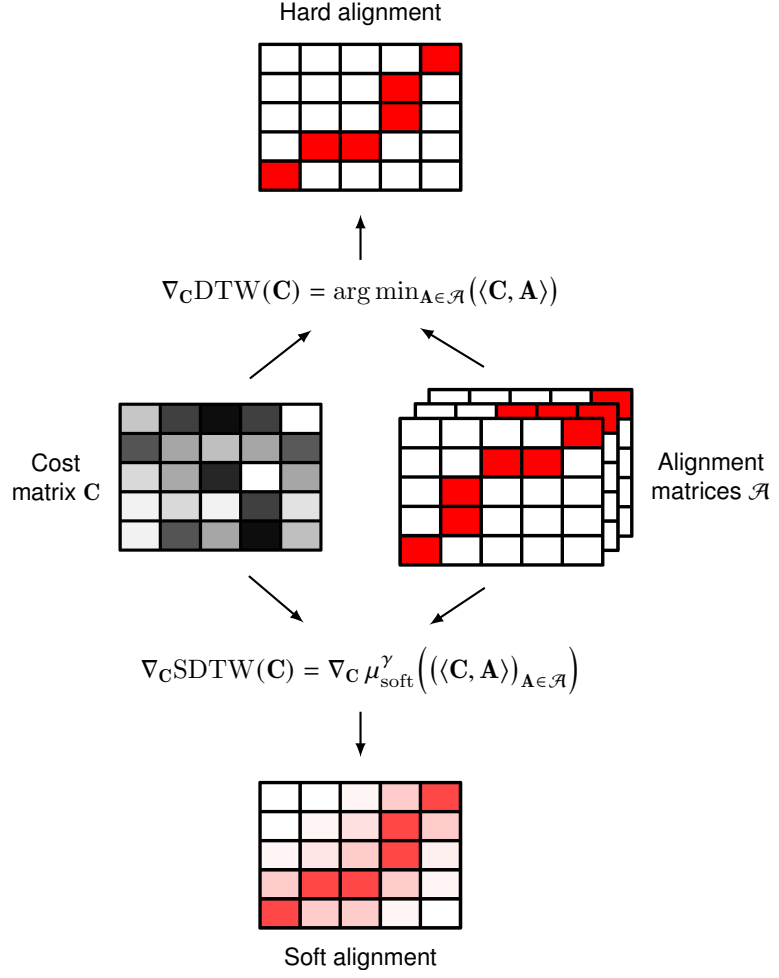
$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} = -\gamma \frac{1}{Z(\mathbf{C})} \frac{\partial Z(\mathbf{C})}{\partial \mathbf{C}(n, m)}. \quad (4.13)$$

Calculating the derivative of the partition function then gives:

$$\frac{\partial Z(\mathbf{C})}{\partial \mathbf{C}(n, m)} = \sum_{\mathbf{A} \in \mathcal{A}} \frac{\partial}{\partial \mathbf{C}(n, m)} e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma} \quad (4.14)$$

$$= -\frac{1}{\gamma} \sum_{\mathbf{A} \in \mathcal{A}} \mathbf{A}(n, m) e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma}. \quad (4.15)$$

Figure 4.3: Illustration of DTW and SDTW gradient computation. DTW returns the alignment matrix of the cost-minimizing path, whereas SDTW returns a sum of all alignment matrices, weighted with their probability of minimizing the alignment cost.



Finally, substituting (4.15) in (4.13) yields:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} = \frac{1}{Z(\mathbf{C})} \cdot \sum_{\mathbf{A} \in \mathcal{A}} \mathbf{A}(n, m) e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma} \quad (4.16)$$

$$= \sum_{\mathbf{A} \in \mathcal{A}} \mathbf{A}(n, m) \cdot \frac{e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma}}{Z(\mathbf{C})}. \quad (4.17)$$

This reformulation allows us to interpret the gradient $\nabla_C \text{SDTW}(\mathbf{C})$ in a probabilistic way. By defining the probability of an alignment $\mathbf{A}$ (given a temperature parameter γ) as

$$p(\mathbf{A}|\mathbf{C}) := \frac{e^{-\langle \mathbf{A}, \mathbf{C} \rangle / \gamma}}{Z(\mathbf{C})}, \quad (4.18)$$

we can reformulate the SDTW gradient as the expectation over all alignments $\mathbf{A}$ for a given cost matrix $\mathbf{C}$:

$$\nabla_{\mathbf{C}} \text{SDTW}(\mathbf{C}) = \sum_{\mathbf{A} \in \mathcal{A}} p(\mathbf{A}|\mathbf{C}) \mathbf{A} \quad (4.19)$$

$$= \mathbb{E}_{\mathbf{A} \sim p(\mathbf{A}|\mathbf{C})} [\mathbf{A}] \quad (4.20)$$

$$=: \mathbf{H}, \quad (4.21)$$

where we call $\mathbf{H} \in \mathbb{R}^{N \times M}$ the gradient matrix.

4.4 SDTW: Local Formulation

As in classical DTW, directly evaluating the global SDTW formulation, i.e., summing over all valid alignment paths in $\mathcal{A}$ (see Section 4.3), is computationally infeasible in practice. Instead, and analogously to DTW, the SDTW cost and gradient can be computed efficiently using a DP algorithm [30].

Forward Pass: SDTW Cost

The SDTW forward pass recursively constructs an accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$. The recursion is initialized at the boundaries as

$$\mathbf{D}(n, 1) = \sum_{i=1}^n \mathbf{C}(i, 1) \quad \text{for } n \in [1 : N] \quad (4.22)$$

$$\mathbf{D}(1, m) = \sum_{j=1}^m \mathbf{C}(1, j) \quad \text{for } m \in [1 : M]. \quad (4.23)$$

For all interior points with $n \in [2 : N]$ and $m \in [2 : M]$, the recursion proceeds in the same manner as in DTW, except that the hard minimum operator is replaced by the softmin function:

$$\mathbf{D}(n, m) = \mathbf{C}(n, m) + \mu_{\text{soft}}^{\gamma}(\mathbf{D}(n-1, m), \mathbf{D}(n, m-1), \mathbf{D}(n-1, m-1)). \quad (4.24)$$

The resulting SDTW cost obtained via DP, denoted $\text{SDTW}^{\text{DP}}(\mathbf{C})$, is then given by the final entry of the accumulated cost matrix:

$$\text{SDTW}^{\text{DP}}(\mathbf{C}) = \mathbf{D}(N, M). \quad (4.25)$$

As shown in prior work [30, 94] and further discussed in Chapter 10, the global SDTW formulation $\text{SDTW}(\mathbf{C})$ (see (4.8)) is equivalent to the local, recursion-based cost $\text{SDTW}^{\text{DP}}(\mathbf{C})$ provided that the recursion in (4.24) employs the softmin operator. Identical to classical DTW, the recursion has complexity $O(NM)$ in time and space. A summary of the SDTW forward algorithm is provided in Algorithm 4.1.

Algorithm 4.1 SDTW forward pass

```

1: Inputs: cost matrix  $\mathbf{C} \in \mathbb{R}^{N \times M}$ , temperature  $\gamma \in \mathbb{R}$ 
2: for  $m = 1, \dots, M$  do
3:    $\mathbf{D}(1, m) = \sum_{j=1}^m \mathbf{C}(1, j)$ 
4: end for
5: for  $n = 1, \dots, N$  do
6:    $\mathbf{D}(n, 1) = \sum_{i=1}^n \mathbf{C}(i, 1)$ 
7: end for
8: for  $m = 2, \dots, M$  do
9:   for  $n = 2, \dots, N$  do
10:     $\mathbf{D}(n, m) = \mathbf{C}(n, m) + \mu_{\text{soft}}^\gamma(\mathbf{D}(n-1, m), \mathbf{D}(n, m-1), \mathbf{D}(n-1, m-1))$ 
11:   end for
12: end for
13: Output: SDTW cost  $\mathbf{D}(N, M) \in \mathbb{R}$ 

```

Backward Pass: SDTW Gradient

In the following section, we describe the calculation of the SDTW gradient $\mathbf{H}(n, m) := \partial \text{SDTW}(\mathbf{C}) / \partial \mathbf{C}(n, m)$ with a DP recursion, following [30]. We start by writing out the derivative using the chain rule:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} = \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{C}(n, m)} \quad (4.26)$$

$$= \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)} \cdot \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} \quad (4.27)$$

$$= \mathbf{E}(n, m) \cdot \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)}, \quad (4.28)$$

where we have defined the shorthand notation

$$\mathbf{E}(n, m) := \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)}. \quad (4.29)$$

Again applying the chain rule, we split the derivative $\partial \mathbf{D}(N, M) / \partial \mathbf{D}(n, m)$ into the contributions from the three steps $(n+1, m)$, $(n, m+1)$, $(n+1, m+1)$ that directly follow after cell (n, m) :

$$\begin{aligned} \mathbf{E}(n, m) &= \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)} = \\ &= \underbrace{\frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n+1, m)} \cdot \frac{\partial \mathbf{D}(n+1, m)}{\partial \mathbf{D}(n, m)}}_{=\mathbf{E}(n+1, m) \cdot \mathbf{B}^{(1,0)}(n, m)} + \underbrace{\frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m+1)} \cdot \frac{\partial \mathbf{D}(n, m+1)}{\partial \mathbf{D}(n, m)}}_{=\mathbf{E}(n, m+1) \cdot \mathbf{B}^{(0,1)}(n, m)} + \underbrace{\frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n+1, m+1)} \cdot \frac{\partial \mathbf{D}(n+1, m+1)}{\partial \mathbf{D}(n, m)}}_{=\mathbf{E}(n+1, m+1) \cdot \mathbf{B}^{(1,1)}(n, m)}, \end{aligned} \quad (4.30)$$

with

$$\mathbf{B}^{(i,j)}(n, m) := \frac{\partial \mathbf{D}(n+i, m+j)}{\partial \mathbf{D}(n, m)}. \quad (4.31)$$

Thus, we can simplify (4.30) to

$$\mathbf{E}(n, m) = \mathbf{E}(n+1, m) \mathbf{B}^{(1,0)}(n, m) + \mathbf{E}(n, m+1) \mathbf{B}^{(0,1)}(n, m) + \mathbf{E}(n+1, m+1) \mathbf{B}^{(1,1)}(n, m), \quad (4.32)$$

which is the core of the SDTW backtracking recursion. Next, we calculate the terms $\mathbf{B}^{(i,j)}(n, m)$, giving $\mathbf{B}^{(1,0)}(n, m)$ as an example:

$$\mathbf{B}^{(1,0)}(n, m) = \frac{\partial \mathbf{D}(n+1, m)}{\partial \mathbf{D}(n, m)} \quad (4.33)$$

$$= \frac{\partial}{\partial \mathbf{D}(n, m)} \left(\mathbf{C}(n+1, m) + \mu_{\text{soft}}^{\gamma}(\mathbf{D}(n, m), \mathbf{D}(n+1, m-1), \mathbf{D}(n, m-1)) \right) \quad (4.34)$$

$$= \frac{\partial}{\partial \mathbf{D}(n, m)} \mu_{\text{soft}}^{\gamma}(\mathbf{D}(n, m), \mathbf{D}(n+1, m-1), \mathbf{D}(n, m-1)) \quad (4.35)$$

$$= -\gamma \frac{\partial}{\partial \mathbf{D}(n, m)} \log \left(e^{-\mathbf{D}(n, m)/\gamma} + e^{-\mathbf{D}(n+1, m-1)/\gamma} + e^{-\mathbf{D}(n, m-1)/\gamma} \right) \quad (4.36)$$

$$= \frac{e^{-\mathbf{D}(n, m)/\gamma}}{e^{-\mathbf{D}(n, m)/\gamma} + e^{-\mathbf{D}(n+1, m-1)/\gamma} + e^{-\mathbf{D}(n, m-1)/\gamma}}. \quad (4.37)$$

By definition of the forward recursion, we have

$$\mathbf{D}(n+1, m) = \mathbf{C}(n+1, m) + \mu_{\text{soft}}^{\gamma}(\mathbf{D}(n, m), \mathbf{D}(n+1, m-1), \mathbf{D}(n, m-1)) \quad (4.38)$$

$$= \mathbf{C}(n+1, m) - \gamma \log \left(e^{-\mathbf{D}(n, m)/\gamma} + e^{-\mathbf{D}(n+1, m-1)/\gamma} + e^{-\mathbf{D}(n, m-1)/\gamma} \right), \quad (4.39)$$

which we re-order to:

$$\gamma \log \left(e^{-\mathbf{D}(n, m)/\gamma} + e^{-\mathbf{D}(n+1, m-1)/\gamma} + e^{-\mathbf{D}(n, m-1)/\gamma} \right) = \mathbf{C}(n+1, m) - \mathbf{D}(n+1, m). \quad (4.40)$$

Dividing by both sides by γ and applying the exponential function, we obtain

$$e^{-\mathbf{D}(n, m)/\gamma} + e^{-\mathbf{D}(n+1, m-1)/\gamma} + e^{-\mathbf{D}(n, m-1)/\gamma} = e^{(\mathbf{C}(n+1, m) - \mathbf{D}(n+1, m))/\gamma}, \quad (4.41)$$

which finally gives a compact expression for $\mathbf{B}^{(1,0)}(n, m)$:

$$\mathbf{B}^{(1,0)}(n, m) = \frac{e^{-\mathbf{D}(n, m)/\gamma}}{e^{(\mathbf{C}(n+1, m) - \mathbf{D}(n+1, m))/\gamma}} \quad (4.42)$$

$$= e^{(\mathbf{D}(n+1, m) - \mathbf{C}(n+1, m) - \mathbf{D}(n, m))/\gamma}. \quad (4.43)$$

Similarly, we obtain the other two terms $\mathbf{B}^{(0,1)}(n, m)$ and $\mathbf{B}^{(1,1)}(n, m)$ as:

$$\mathbf{B}^{(0,1)}(n, m) = e^{(\mathbf{D}(n, m+1) - \mathbf{C}(n, m+1) - \mathbf{D}(n, m))/\gamma} \quad (4.44)$$

$$\mathbf{B}^{(1,1)}(n, m) = e^{(\mathbf{D}(n+1, m+1) - \mathbf{C}(n+1, m+1) - \mathbf{D}(n, m))/\gamma}. \quad (4.45)$$

Algorithm 4.2 SDTW backward pass

```

1: Inputs:  $\mathbf{C}, \mathbf{D}, \gamma$ 
2:  $\mathbf{C}(N+1, \cdot) = \mathbf{C}(\cdot, M+1) = 0$ 
3:  $\mathbf{D}(N+1, \cdot) = \mathbf{D}(\cdot, M+1) = -\infty$ 
4:  $\mathbf{D}(N+1, M+1) = \mathbf{D}(N, M)$ 
5:  $\mathbf{E}(N+1, \cdot) = \mathbf{E}(\cdot, M+1) = 0$ 
6:  $\mathbf{E}(N+1, M+1) = 1$ 
7: for  $m = M, \dots, 1$  do
8:   for  $n = N, \dots, 1$  do
9:      $\mathbf{B}^{(1,0)}(n, m) = \exp((\mathbf{D}(n+1, m) - \mathbf{C}(n+1, m) - \mathbf{D}(n, m)) / \gamma)$ 
10:     $\mathbf{B}^{(0,1)}(n, m) = \exp((\mathbf{D}(n, m+1) - \mathbf{C}(n, m+1) - \mathbf{D}(n, m)) / \gamma)$ 
11:     $\mathbf{B}^{(1,1)}(n, m) = \exp((\mathbf{D}(n+1, m+1) - \mathbf{C}(n+1, m+1) - \mathbf{D}(n, m)) / \gamma)$ 
12:     $\mathbf{E}(n, m) = \mathbf{E}(n+1, m)\mathbf{B}^{(1,0)}(n, m) + \mathbf{E}(n, m+1)\mathbf{B}^{(0,1)}(n, m) + \mathbf{E}(n+1, m+1)\mathbf{B}^{(1,1)}(n, m)$ 
13:   end for
14: end for
15: Output: gradient matrix  $\mathbf{H} = \mathbf{E}$ 

```

As the derivative of a cell in the accumulated cost matrix w.r.t. the same cell in the local cost matrix is exactly one for equal step weights:

$$\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} = 1, \quad (4.46)$$

we can substitute (4.32) in (4.30) and obtain the SDTW gradient $\mathbf{H}$ as:

$$\mathbf{H}(n, m) = \mathbf{E}(n, m) \cdot \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} \quad (4.47)$$

$$= \mathbf{E}(n, m). \quad (4.48)$$

An overview of the complete algorithm is provided in Algorithm 4.2. With slight abuse of notation, we initialize the backward recursion by setting values to the $(N+1)$ -th and $(M+1)$ -th columns and rows of $\mathbf{C}$, $\mathbf{D}$, and $\mathbf{E}$. It can be shown that these settings correctly initialize the algorithm (see [30]).

Interpretation of the SDTW Backward Algorithm

The backward pass of SDTW yields a matrix $\mathbf{E} \in [0, 1]^{N \times M}$, which can be interpreted as a *soft alignment matrix*. Each entry $\mathbf{E}(n, m)$ represents the probability that the cell (n, m) is part of the cost-minimizing alignment path. The matrix $\mathbf{E}$ is computed recursively in reverse order (compared to the forward pass), starting from the end cell (N, M) . Since every valid alignment path must end at this final cell, the boundary condition gives the starting point of the recursion as $\mathbf{E}(N, M) = 1$.

The backward recursion relies on coefficients $\mathbf{B}^{(i,j)}(n, m) \in [0, 1]$, which can be interpreted as *soft backtracking pointers*. Specifically, $\mathbf{B}^{(i,j)}(n, m)$ denotes the probability that the transition from cell (n, m) to cell $(n+i, m+j)$ was the cost-minimizing step in the forward recursion.

This mechanism is directly analogous to backtracking in classical DTW. In the DTW setting, a single optimal alignment path is recovered by starting at (N, M) and iteratively following deterministic backpointers that indicate the minimum-cost predecessor at each step. In contrast, SDTW replaces these hard backpointers with probabilistic transition weights $\mathbf{B}$ and replaces a single alignment path with a distribution over paths $\mathbf{E}$. As a result, the backward algorithm propagates alignment probabilities rather than selecting a single optimal trajectory, yielding the soft alignment matrix $\mathbf{E}$. In the considered case of equal step weights for the vertical, horizontal, and diagonal alignment step, the soft alignment matrix is equal to the gradient matrix, i.e., $\mathbf{E} = \mathbf{H}$. We refer to Section 6.2 for a discussion of the relation between $\mathbf{H}$ and $\mathbf{E}$ for variable step weights.

4.5 Limit Cases

In the following section, we will investigate the asymptotic behavior of the SDTW loss for $\gamma \rightarrow 0$ and $\gamma \rightarrow \infty$. We refer to Section 4.2 and Appendix B for further background on the behavior of the softmin function for limit cases.

Asymptotic Behavior of SDTW for $\gamma \rightarrow 0$

As the softmin function approximates the hard minimum function for $\gamma \rightarrow 0$, the SDTW cost function approximates classical DTW in this limit case:

$$\lim_{\gamma \rightarrow 0} \text{SDTW}(\mathbf{C}) = \min_{\mathbf{A} \in \mathcal{A}} \langle \mathbf{A}, \mathbf{C} \rangle \quad (4.49)$$

$$= \text{DTW}(\mathbf{C}) . \quad (4.50)$$

The derivative of the softmin function in the limit case $\gamma \rightarrow 0$ becomes a uniform distribution over its minimizers (see Section 4.2). Transferring this observation to the gradient of the SDTW function in (4.19), and defining the minimum alignment cost $c_{\min} := \min_{\mathbf{A} \in \mathcal{A}} \langle \mathbf{A}, \mathbf{C} \rangle$ as well as the set of cost-minimizing alignment paths $\mathcal{A}^* = \{\mathbf{A} \in \mathcal{A} \mid \langle \mathbf{A}, \mathbf{C} \rangle = c_{\min}\}$, we obtain the SDTW gradient as the average over the cost-minimizing alignment paths:

$$\lim_{\gamma \rightarrow 0} \nabla_{\mathbf{C}} \text{SDTW}(\mathbf{C}) = \frac{1}{|\mathcal{A}^*|} \sum_{\mathbf{A} \in \mathcal{A}^*} \mathbf{A} . \quad (4.51)$$

In case there exists a single cost-minimizing alignment path $\mathbf{A}^*$, the SDTW gradient is equal to this particular alignment matrix.

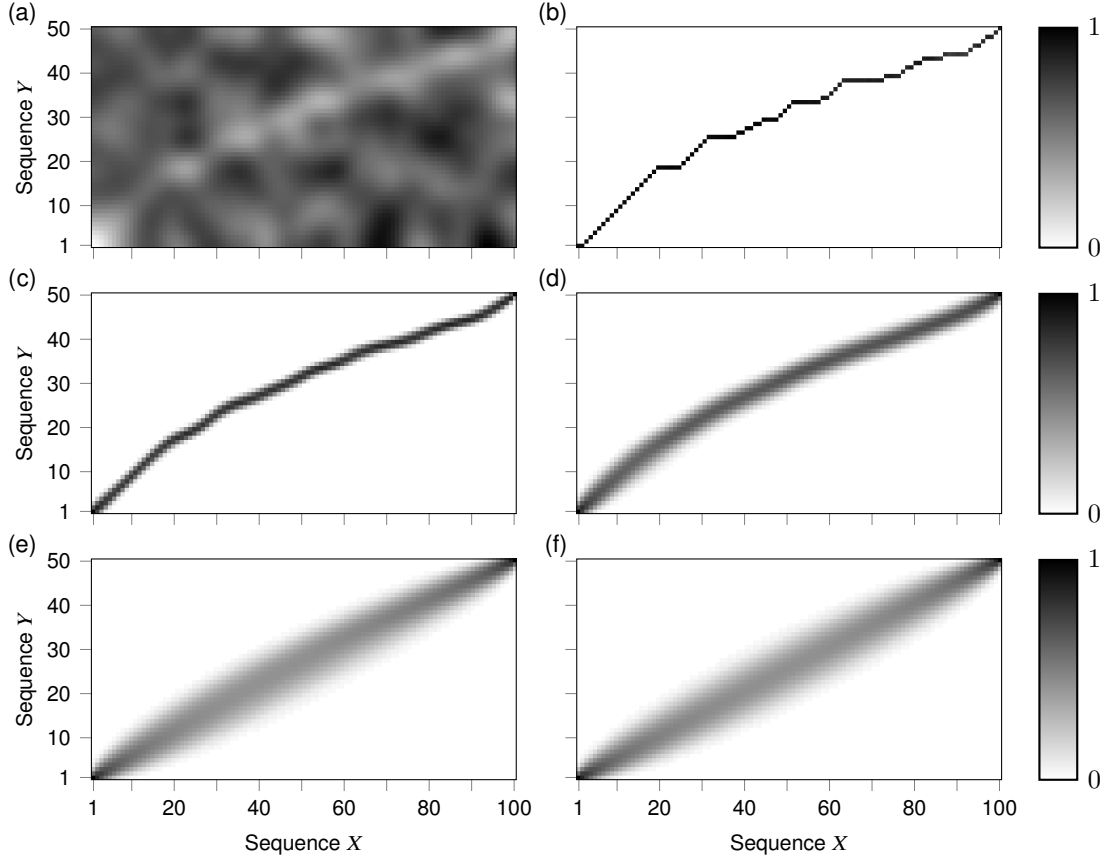


Figure 4.4: Illustration of (a) cost matrix and SDTW gradient $\mathbf{H}$ or soft alignment matrices $\mathbf{E}$ (in our case, we have $\mathbf{H} = \mathbf{E}$) for different temperature parameters γ . (b) $\gamma = 0$. (c) $\gamma = 0.1$. (d) $\gamma = 1$. (e) $\gamma = 10$. (f) $\gamma = 100$.

Asymptotic Behavior of SDTW for $\gamma \rightarrow \infty$

In contrast, when γ takes large values, the softmin function transitions from a minimum-like operator to a mean-like function, as described in Section 4.2. Thus, we obtain the behavior of the SDTW function for $\gamma \rightarrow \infty$ as the mean over all valid alignments minus a constant:

$$\text{SDTW}(\mathbf{C}) \xrightarrow{\gamma \rightarrow \infty} \frac{1}{|\mathcal{A}|} \left(\sum_{\mathbf{A} \in \mathcal{A}} \langle \mathbf{A}, \mathbf{C} \rangle \right) - \gamma \log |\mathcal{A}|. \quad (4.52)$$

For a probabilistic interpretation of (4.52), assuming that all alignments have equal probabilities, let $p^{(n,m)}$ denote the probability the cell (n, m) is part of a valid alignment path $\mathbf{A} \in \mathcal{A}$:

$$p^{(n,m)} = \frac{1}{|\mathcal{A}|} \sum_{\mathbf{A} \in \mathcal{A}} \mathbf{A}(n, m). \quad (4.53)$$

Analogously, $p^{(n,m)}$ can be thought of as the ratio of all paths in $\mathcal{A}$ that align sequence elements (n, m) . Then, with the set $\mathcal{I} = [1 : N] \times [1 : M]$ of all cells of the cost matrix, the SDTW cost for $\gamma \rightarrow \infty$

becomes the sum of all cells of the cost matrix $\mathbf{C}$, weighted with their probability to be part of a valid alignment path:

$$\text{SDTW}(\mathbf{C}) \xrightarrow{\gamma \rightarrow \infty} \left(\sum_{(n,m) \in \mathcal{I}} \mathbf{C}(n,m) \cdot p^{(n,m)} \right) - \gamma \log |\mathcal{A}|. \quad (4.54)$$

For $\gamma \rightarrow \infty$ the derivative of the softmin function can be approximated by a uniform distribution (see Section 4.2). Using this knowledge for the calculation of the SDTW gradient in (4.19), we obtain for the limit case $\gamma \rightarrow \infty$:

$$\lim_{\gamma \rightarrow \infty} \nabla_{\mathbf{C}} \text{SDTW}(\mathbf{C}) = \frac{1}{|\mathcal{A}|} \sum_{\mathbf{A} \in \mathcal{A}} \mathbf{A}. \quad (4.55)$$

That is, the SDTW gradient for $\gamma \rightarrow \infty$ degenerates to an averaging over all valid alignment paths, i.e., the value of a gradient cell (n, m) denotes the probability that a valid alignment path passes through that cell, assuming a uniform distribution over paths and ignoring the local cost $\mathbf{C}$.

Discussion

In Figure 4.4, we visualize the SDTW gradient $\mathbf{H}$ (which is equal to the soft alignment matrix $\mathbf{E}$ for uniform step weights) for different values of γ . In Figure 4.4b, we observe that for $\gamma \rightarrow 0$, the gradient is indeed equal to a single cost-minimizing alignment matrix. By slightly increasing γ to 0.1 (Figure 4.4c) and 1.0 (Figure 4.4d), the alignment path becomes more and more blurred, while still following the original trajectory. In the limit case $\gamma \rightarrow \infty$, the SDTW gradient becomes independent of the input cost matrix and degenerates to a probability distribution over all valid alignment paths, with probability mass mainly concentrated around the main diagonal (see Figure 4.4f).

A Note on Optimizing the Temperature Parameter γ

At first glance, it may seem natural to treat the temperature parameter γ as a trainable parameter and optimize it by minimizing the SDTW cost. However, note that the approximation of the SDTW objective for $\gamma \rightarrow \infty$ contains a negative offset that scales linearly with γ , as written in (4.52) and (4.54). While this offset does not affect the gradient w.r.t. the local cost matrix $\mathbf{C}$ (see (4.55)), it prevents optimizing γ as a trainable parameter, as a larger γ would always reduce the SDTW cost. Therefore, the hyperparameter must be set manually, for example, after inspection of the internally aligned targets, as described in Chapter 8.

5 Stabilizing SDTW for End-to-End Training

This chapter is based on [167]. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Simon Deniffel validated the approaches as part of his research project. Michael Krause shared his expertise in training with the SDTW loss function. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

Although SDTW enables end-to-end learning from weakly aligned data, the soft alignments estimated in the early training stages are often unreliable and can destabilize DNN training. In particular, severe misalignments between unrelated input and target frames may dominate the alignment, causing misleading gradients and poor convergence.

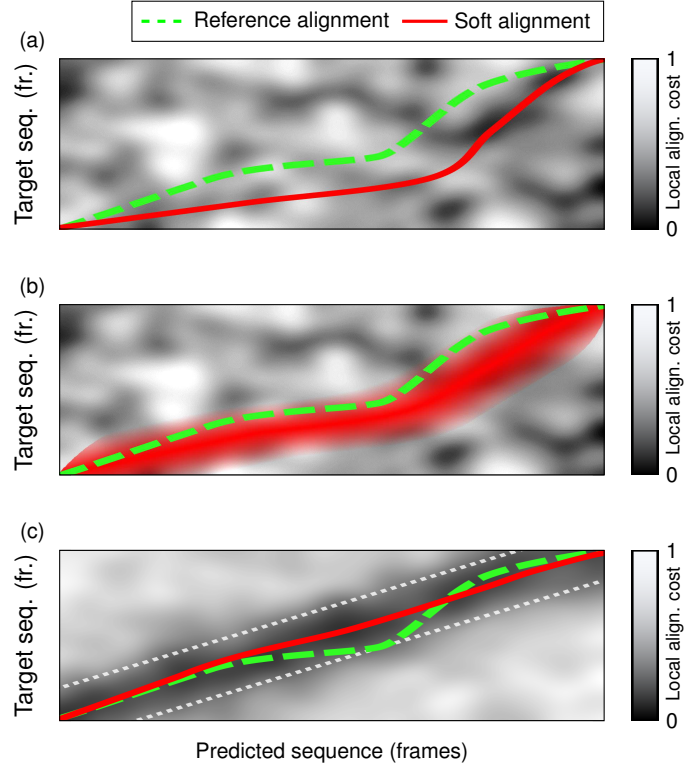
This chapter introduces three complementary stabilization strategies applied to the SDTW input representations or hyperparameters, namely a temperature scheduling, diagonal priors, and sequence unfolding. Intermediate alignments are stabilized and guided towards plausible regions, while the local flexibility of SDTW is preserved. We validate the effectiveness of these strategies in a training task for PCE from weakly aligned data, demonstrating that stable optimization is achievable without sacrificing alignment flexibility and prediction performance.

The remainder of this chapter is structured as follows. First, in Section 5.1, we discuss the problem of unstable training under the SDTW loss. Next, in Section 5.2, we introduce three conceptually different strategies for stabilizing DNN with SDTW. After describing the experimental setup in Section 5.3, we evaluate cause and effect of training problems with SDTW in Section 5.4, along with the impact of our stabilizing strategies. Finally, we conclude with Section 5.5.

5.1 Motivation and Related Work

While SDTW has been applied in prior work for tasks such as performance–score synchronization [1] or for training DNNs for MPE with weakly aligned pitch annotations [83], experiments indicated training

Figure 5.1: Deviation of strong reference alignments (dashed green) and soft alignments (red) and stabilizing strategies. (a) Alignment mismatch of standard SDTW. Stabilizing alignments with (b) hyperparameter scheduling and (c) diagonal prior.



instabilities with SDTW when the sequence lengths of inputs and targets are significantly different [83]. This poses a severe problem in many MIR tasks, where sequences of input audio are typically very long, while sequences of weakly aligned targets (i.e., without encoded note durations) are significantly shorter.

In this chapter, we investigate the cause of training instabilities under the SDTW loss and show that it is due to a mismatch between the estimated soft alignment and the reference alignment (see Figure 5.1a) in the early stages of training. This mismatch causes incorrect parameter updates and the training may diverge. Therefore, we introduce and investigate strategies to decrease this alignment error to stabilize training. In particular, we analyze a hyperparameter scheduling strategy to yield smooth alignments in the early training phase (see Figure 5.1b) as well as the strategy of adding a diagonal prior to the SDTW cost matrix to initially favor diagonal alignments (see Figure 5.1c). Furthermore, we investigate a sequence unfolding approach, where we uniformly stretch the weak target sequence to the length of the input sequence as proposed in [83]. We choose DNN-based PCE as an exemplary task to study the training process of standard SDTW and the impact of our stabilizing strategies. We demonstrate that the hyperparameter scheduling and the diagonal prior strategies reliably reduce label mismatch in the early training stage and therefore lead to successful trainings. In addition, these two strategies are computationally efficient and require only small modifications to the standard SDTW algorithm.

5.2 Stabilizing Training with SDTW

In this section, we introduce three strategies for stabilizing SDTW-based training: hyperparameter scheduling, diagonal prior, and sequence unfolding.

5.2.1 Hyperparameter Scheduling

As described in Section 4.2, the softmin temperature parameter γ controls the smoothness of the SDTW soft alignments. While a low value of γ is desirable to ensure exact correspondences between predictions and targets due to sharp alignments, the latter are problematic in the initial training phase as inaccurate predictions from randomly initialized network parameters lead to erroneous alignments, thus hampering convergence. Therefore, as a first strategy to stabilize SDTW training, we discuss an epoch-dependent scheduling of γ . Starting a training with a large softmin temperature $\gamma^{\text{start}} = 10$ makes the soft alignment fuzzier, which leads to coarse, yet mostly meaningful target assignments (see Figure 5.1b). After ten epochs with $\gamma = 10$, when the trained network predicts meaningful features, we linearly reduce γ during the following ten epochs to a final value of $\gamma^{\text{final}} = 0.1$, which stays constant for the remaining training.

5.2.2 Diagonal Prior

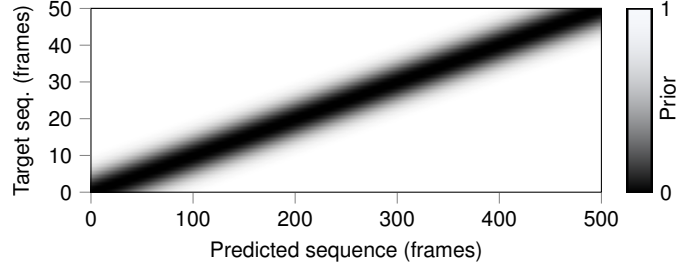
On average, the correct alignment of two sequences with arbitrary symbol durations has a higher probability to be close to the diagonal than to deviate from it. Therefore, as a second approach to stabilize the initial training phase, we investigate an additive prior $\mathbf{P} \in \mathbb{R}^{N \times M}$ which penalizes elements of the cost matrix $\mathbf{C}$ that are far from the diagonal (see Figure 5.2 for an illustration of a prior matrix). A similar strategy was employed in [133] for restricting speech-text alignments to the diagonal. Assuming equal symbol durations, the diagonal alignment of a target $\mathbf{y}_m$ starts at input frame $q_m = \lfloor \frac{Nm}{M} \rfloor$ and ends at $q_{m+1} - 1$. To yield no penalty along the diagonal and a smoothly increasing penalty for distant alignments, we define the elements of the prior matrix as

$$\mathbf{P}(n, m) = 1 - \begin{cases} 1, & q_m \leq n < q_{m+1} \\ \exp\left(\frac{(n-q_m)^2}{-2\nu}\right), & n < q_m \\ \exp\left(\frac{(n-q_{m+1})^2}{-2\nu}\right), & n \geq q_{m+1}, \end{cases} \quad (5.1)$$

where the parameter ν controls the sharpness of the prior. In our experiments, we use $\nu = 1000$. Finally, the prior matrix is added to the cost matrix with a weight ω to obtain the penalized cost matrix

$$\mathbf{C}_p := \mathbf{C} + \omega \mathbf{P}, \quad (5.2)$$

Figure 5.2: Diagonal prior matrix $\mathbf{P}$ for $N = 500$, $M = 50$ and $\nu = 1000$.



which replaces $\mathbf{C}$ for the SDTW cost computation. Similarly to the hyperparameter scheduling strategy, we choose a constant prior weight $\omega = 3$ during the first five epochs and then linearly reduce it to $\omega = 0$ during the following five epochs.

Note that the numerical parameters for the strategies presented in Sections 5.2.1 and 5.2.2 were determined empirically by the authors and small changes did not affect the training performance. However, when training on sequences of different length, with a different learning rate, or other DNN types, parameters should be adjusted on a validation set. As presented in Section 5.4, analysis of the soft alignment matrix $\mathbf{E}$ provides a good indication of the current alignment stability.

5.2.3 Sequence Unfolding

Based on the observation that equal sequence lengths stabilize SDTW training, a third strategy is to uniformly unfold the target sequence (see also [83]). The unfolded target sequence $Y^U = \{y_0^U, y_1^U, \dots, y_{N-1}^U\}$ is constructed by uniformly repeating elements from the weakly aligned target sequence, i.e., setting

$$y_n^U \leftarrow y_{\lfloor \frac{Mn}{N} \rfloor} \quad (5.3)$$

to yield equal sequence lengths of the predictions X and the targets Y^U . Note that the repetition of target vectors introduces ambiguities, leading to multiple optimum alignments.

5.3 Experimental Setup

In this section, we describe the task for our case study, the employed dataset, as well as the used DNN architecture and the training procedure.

5.3.1 Task: Pitch Class Estimation

We choose PCE from music recordings as an illustrative case study to investigate the problems of the SDTW loss function and the effect of the stabilizing strategies. In our experimental setting, a DNN takes N frames of input audio (including context) and, for all frames, predicts twelve-dimensional pitch class

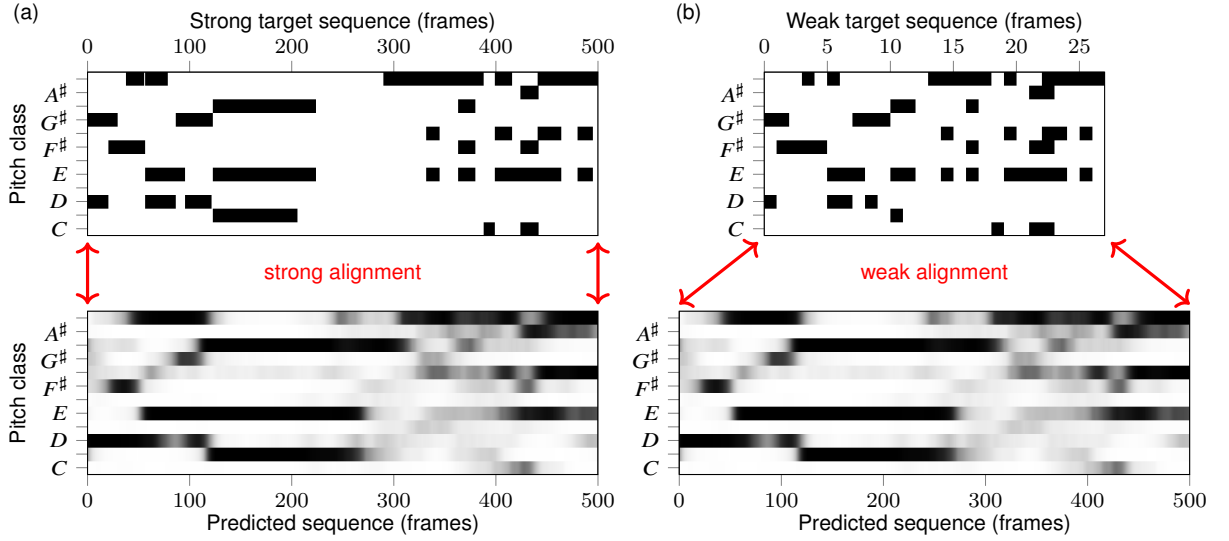


Figure 5.3: Alignment between training targets and predicted pitch class features X for the running example from *Frühlingstraum*. (a) Strong reference alignment for MSE loss with strong targets Y^S . (b) Weak alignment for SDTW loss with weak targets Y .

activation vectors X (see Figure 5.3 for an illustration of predicted pitch class features). We want to train the DNN such that the predictions X match the training targets as close as possible. In the case of strong targets Y^S , each predicted frame x_n is assigned to exactly one target frame y_n^S using a strong alignment (see Figure 5.3a). When using weak targets Y , SDTW internally computes a soft alignment based on the cost matrix C to assign predictions and targets (see Figure 5.3b).

5.3.2 Dataset

Throughout all experiments, we use the Schubert Winterreise Dataset (SWD) [153] which contains audio recordings and strongly aligned pitch class annotations. Winterreise is a song cycle for piano and singer, consisting of 24 songs. For each song, SWD comprises nine different performances, resulting in $9 \cdot 24$ recorded songs with a total duration of 10 h 50 min. We split the dataset for training, validation, and testing using a performance split [153]. The publicly available performances by Huesch (HU33, recorded in 1933) and Scarlata (SC06, recorded in 2006) were annotated manually [153] and constitute the test set. For training and evaluation we choose sequences of length $N = 500$, corresponding to approximately 8.7 s of audio at a sampling rate of 22 050 Hz and a hop length of 384 samples. In order to generate weak training targets Y from SWD (which provides strongly aligned pitch class annotations Y^S , see Figure 5.3a), we remove all adjacent repetitions of a pitch class vector (see Figure 5.3b) [146]. We choose an excerpt from the song *Frühlingstraum*, performed by Randall Scarlata (SC06), as a running example (see Figure 5.3) to visualize the soft alignment matrices (see Figure 5.4).

Table 5.1: Musically motivated CNN architecture [146, 154].

Layer	Kernel Size	Stride	Output Shape
Prefiltering			
LayerNorm			$(N + 74, 216, 5)$
Conv2D	15×15	(1,1)	$(N + 74, 216, 20)$
MaxPool	3×1	(1,1)	$(N + 74, 216, 20)$
Dropout			
Binning to MIDI pitches			
Conv2D	3×3	(1,3)	$(N + 74, 72, 20)$
MaxPool	13×1	(1,1)	$(N + 74, 72, 20)$
Dropout			
Time reduction			
Conv2D	75×1	(1,1)	$(N, 72, 10)$
Dropout			
Chroma reduction			
Conv2D	1×1	(1,1)	$(N, 72, 1)$
Dropout			
Conv2D	1×61	(1,12)	$(N, 12, 1)$

5.3.3 DNN Architecture and Training

We adapt a conceptually simple and musically motivated five-layer Convolutional Neural Network (CNN) from [146, 154] with 43383 trainable parameters to predict twelve-dimensional pitch class activation vectors from an input sequence. Table 5.1 provides an overview of the architecture. We choose the Harmonic Constant-Q Transform (HCQT) [14] with five harmonics as an audio feature representation, spanning six octaves at a resolution of three bins per semitone (resulting in 216 frequency bins starting from C1), a hop length of 384 samples and a frame rate of 57.4 Hz. From an input sequence of length $N + 74$, the CNN sequentially predicts N vectors of pitch class activations. For the prediction of one frame, the CNN’s receptive field covers 37 adjacent context frames on each side. Leaky ReLU with a negative slope of 0.3 is used as a nonlinearity after all hidden convolutional layers and sigmoid activation is used after the final layer. The dropout rate is set to 0.2. All models are trained using the Adam optimizer [74] with a batch size of 32 and an initial learning rate of 0.001. We reduce the learning rate by a factor of two if the validation loss did not decrease during the last four epochs, and terminate the training if the validation loss did not decrease during the last twelve epochs. At the end of training, the model from the epoch with the lowest validation loss is restored. The source code for reproducing our experiments, as well as the trained models are available on github.com/groupmm/stabilizing_sdtw.

5.4 Evaluation

In this section, we investigate the training process as well as the prediction accuracy under the standard SDTW loss, and compare it to the discussed stabilizing strategies. For quantitative evaluation, we repeat all DNN trainings ten times from random initializations. For the test set predictions of each trained model, we compute the F-measure w.r.t. time-pitch class bins using a threshold of 0.5. The mean and standard deviation of the F-measures from all trained models are displayed in Table 5.2.

Loss	Targets	γ	Strategy	F-measure	
				mean	std
MSE	strong	-	-	0.82	0.07
SDTW	weak	0.1	-	0.56	0.37
SDTW	weak	0.3	-	0.63	0.32
SDTW	weak	1.0	-	0.24	0.36
SDTW	weak	3.0	-	0.31	0.38
SDTW	weak	10.0	-	0.57	0.37
SDTW	weak	10 $\rightarrow$ 0.1	hyperpar. scheduling	0.80	0.04
SDTW	weak	0.1	diagonal prior	0.81	0.02
SDTW	weak	0.1	sequence unfolding	0.53	0.04

Table 5.2: Averaged test results for DNNs trained on strongly aligned reference targets as well as DNNs trained with SDTW on weakly aligned targets using either the standard configuration or the discussed stabilization strategies. We report the mean (higher is better) and standard deviation (lower is better) of the F-measure.

5.4.1 Baseline: Strongly Aligned Targets

As a first baseline and an upper bound for all following experiments, we consider DNN training with strongly aligned targets Y^S . For the sequence lengths $M = N = 500$ and an MSE loss function, the networks achieve the overall highest mean F-measure of 0.82 with a standard deviation of 0.07 on the test set.

5.4.2 Standard SDTW

We next analyze DNN training with weak targets Y and the unmodified SDTW formulation from [30, 88] as a loss function. We investigate five different values of $\gamma \in \{0.1, \dots, 10\}$ which we keep constant during training. Analyzing the mean F-measure on the test set in Table 5.2, the five variants with standard SDTW yield comparably low results between 0.24 and 0.57, and high standard deviations between 0.32 and 0.38. Between 20 % ($\gamma = 0.3$) and 70 % ($\gamma = 1.0$) of all training runs converged to the all-zero output, indicating a highly unstable training process of standard SDTW. In order to determine the cause of these instabilities, we analyze the quality of automatically generated soft alignments in the SDTW algorithm by visualizing the soft alignment matrix for the running example after training epochs one and 25, respectively. To highlight the effects of small and large values of γ , we focus on the edge cases $\gamma \in \{0.1, 10.0\}$. For $\gamma = 0.1$, the estimated soft alignment exhibits a sharp structure (see Figure 5.4a), which, after a collapse to a single target frame at epoch one, still only marginally overlaps with the reference alignment after 25 epochs. This sharp and erroneous soft alignment causes unstable gradient updates and leads to the collapse of many training runs. When choosing a large softmin temperature $\gamma = 10$, SDTW yields “blurry” soft alignments (see Figure 5.4b) which at least partially capture the actual target frames in early epochs and coincide well with the reference alignments as training progresses. However, a blurry soft alignment also leads to blurry network predictions as multiple target frames are aligned to each predicted frame, thus resulting in a low F-measure when compared to strongly aligned targets..

5.4.3 Stabilizing Strategies

After evaluating the unsatisfactory training behavior of standard SDTW, we investigate the effect of the previously introduced training strategies in the following section. We empirically choose $\gamma = 0.1$ as the final softmin temperature in all following experiments, as sharp alignments are necessary for training an estimator with frame-wise precision.

Hyperparameter Scheduling

First, we combine the advantages of high and low values of γ in a hyperparameter scheduling strategy. Starting a training with $\gamma = 10$, the soft alignment matrix for our running example after one epoch is blurry and at least partially overlapping with the reference alignment (see Figure 5.4c). The successive reduction to $\gamma = 0.1$ until epoch 20 permits sharp alignments at a later training stage. Indeed, Figure 5.4c shows a soft alignment after epoch 25 which is sharp and coincides well with the reference. The mean F-measure (0.80) in Table 5.2, as well as the standard deviation (0.04), are the second best of all SDTW-based trainings. However, as the softmin function is a lower bound for the minimum function which becomes tight for $\gamma \rightarrow 0$ (see Theorem B.1), the SDTW loss is increasing when decreasing γ , despite unchanged network parameters. Therefore, this strategy does not allow for loss-based learning rate scheduling and early stopping before γ is set to its final value.

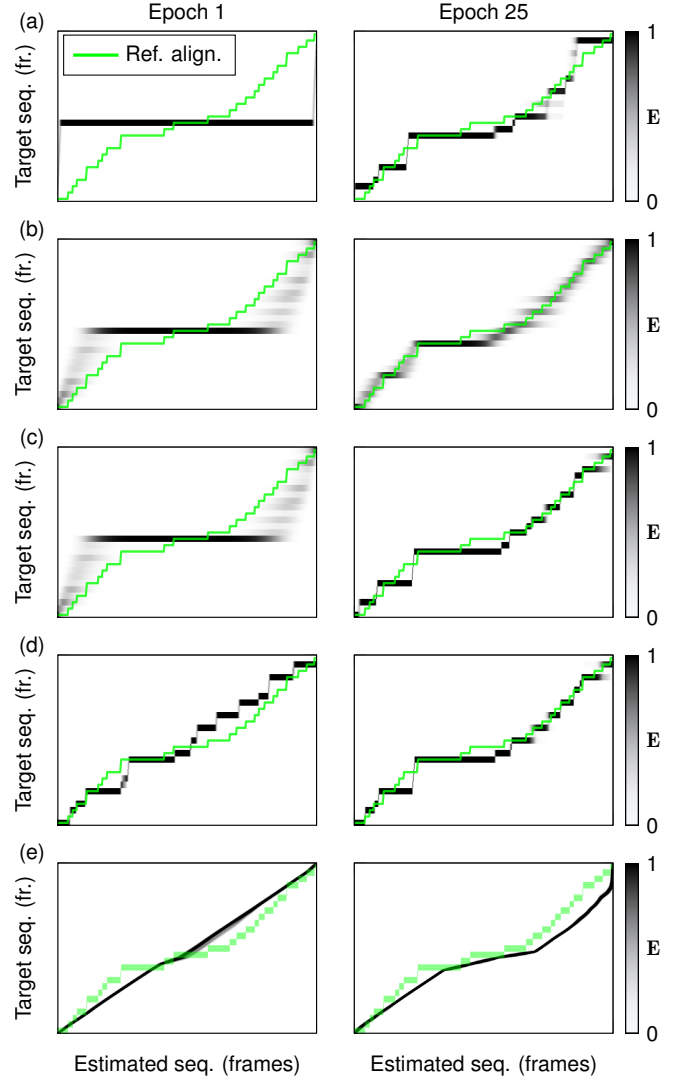
Diagonal Prior

The second strategy stabilizes SDTW trainings with low values of γ by adding a penalty cost to off-diagonal elements of the cost matrix. For our running example in Figure 5.4d, the soft alignment is indeed close to the diagonal after the first training epoch. As, on average, the alignments are diagonal, this often leads to correct assignments of predictions and targets even for randomly initialized DNNs. When the prior weight ω is reduced to zero after the initial training phase, the network is still able to adapt to off-diagonal alignments, as seen in our running example in Figure 5.4d. Analyzing the performance metrics in Table 5.2, using a diagonal prior yields the highest mean F-measure (0.81) and the lowest standard deviation (0.02) of all SDTW variants, almost reaching the mean F-measure of the baseline experiments with strong targets and element-wise MSE loss. Moreover, when the prior weight ω is reduced during training, the loss also decreases and therefore learning rate scheduling and early stopping are possible from the beginning.

Sequence Unfolding

Last, we investigate the strategy of unfolding the weak target sequence to the length of the input, which was employed in [83]. For this strategy, we observe fully diagonal soft alignments in the initial training phase, as visualized for our running example in Figure 5.4e. This is caused by the equal length of the predicted

Figure 5.4: Reference alignment (green) and soft alignment matrix $\mathbf{E}$ (gray/black) for the running example after training epoch 1 (left) and epoch 25 (right) for different training strategies. (a) $\gamma = 0.1$, (b) $\gamma = 10$, (c) hyper-parameter scheduling, (d) diagonal prior, (e) sequence unfolding.



and the target sequence, which can be aligned using only diagonal steps. In the SDTW formulation from [30], the cost of a diagonal step is equal to the cost of a vertical or horizontal step. Thus, for a uniform cost matrix (which is probable at the initial training phase due to random network initialization), taking a diagonal step only accumulates half the cost compared to going “around the corner”, i.e., one step in the vertical and one in the horizontal direction, or vice versa. This diagonalizing behavior leads, on average, to decent soft alignments in the early training phase (as discussed in Section 5.4.3). However, in contrast to the additive diagonal prior strategy, the implicit diagonalization of alignments is not reduced during the training, as can be seen in Figure 5.4e, which still exhibits strong diagonal components after 25 training epochs. Thus, the softly aligned SDTW targets seldom match the reference targets and performance remains low, resulting in a mean F-measure of 0.53 in Table 5.2.

Note that the sequence unfolding strategy adds a significant computational overhead compared to the previous two strategies, as unfolding always corresponds to using a target sequence length of $M = N$. The

forward and backward pass of the SDTW loss function both have linear complexity w.r.t. the sequence lengths $O(MN)$ [30]. Thus, in our setting with $N = 500$ and a mean length of the weak target sequences in the test set of $M = 24$, the unfolding strategy leads to an increase in the computational cost of the SDTW loss by a factor of more than 20.

5.5 Further Notes

In this chapter, we analyzed DNN training instabilities with SDTW as a loss function by the example of PCE. By analysis of the soft alignment matrix, we argued that alignment mismatch in the early training phase often causes a collapse of the training procedure. Motivated by these findings, we investigated three strategies for stabilizing the early training phase. We found that the previously applied strategy of unfolding the weakly aligned target sequence leads to almost exclusively diagonal alignments due to a naïve weighting of horizontal, vertical, and diagonal alignment steps. Furthermore, this strategy is computationally inefficient, as it increases the target sequence length. In contrast, the two introduced strategies of hyperparameter scheduling and diagonal prior can be implemented with negligible additional computational cost and stabilize SDTW-based training by two different mechanisms. The hyperparameter scheduling strategy promotes smooth alignments in the early training phase, which increases the probability of the predicted frame being at least partially aligned to the correct target. Penalizing off-diagonal alignments in the SDTW cost matrix by an additive diagonal prior is a strategy that initially restricts the soft alignment to a region of high probability. Experimental evaluation showed that these strategies reliably stabilize the SDTW training process. Implementing them as a default in the SDTW loss highly increases convergence rates.

6 SDTW with Variable Step Weights

This chapter is based on [168]. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Michael Krause shared his expertise in training with the SDTW loss function. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

Standard DTW and SDTW allow horizontal, vertical, and diagonal warping steps. However, they assign equal cost contribution to all local step sizes, which can encourage pathological alignments, for example by overusing horizontal or vertical moves when sequence lengths differ. In classical DTW, such biases are commonly mitigated by assigning different costs to horizontal, vertical, and diagonal steps.

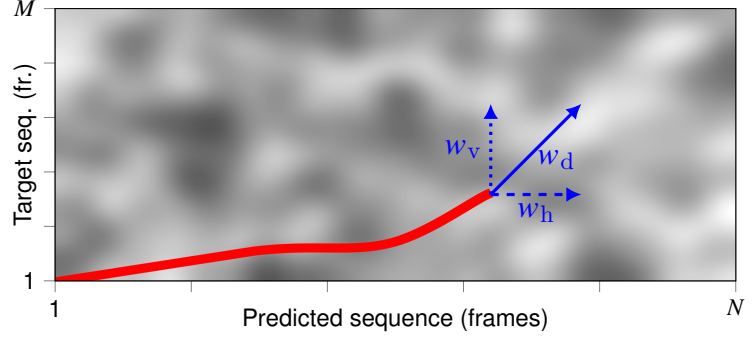
This chapter extends SDTW to support variable step weights and derives efficient DP recursions for both the forward pass and the gradient computation. Experiments demonstrate superior training stability and reduced sensitivity to outliers in learning scenarios with weakly aligned data, showing that an adaption to the structural bias, such as the sequence lengths, is essential for robust differentiable alignment.

The remainder of this chapter is structured as follows. After reviewing related work in Section 6.1, in Section 6.2 we introduce the weighted SDTW formulation and provide DP algorithms for the forward and backward passes. Next, we evaluate our modified SDTW approach in the context of a PCE experiment, which we introduce in Sections 6.3.1 and 6.3.2. In Sections 6.3.3 and 6.3.4, we investigate how reducing the horizontal step weight reduces the influence of outliers and stabilizes the alignments. We conclude in Section 6.4 with an outlook to future applications of the weighted SDTW algorithm.

6.1 Motivation and Related Work

To circumvent stability issues (as observed in Chapter 5) and tailor the alignment computation to a specific problem, the classic DTW algorithm can be extended by assigning individual weights to different alignment step sizes [106], see Figure 6.1 for an illustration. For example, a popular choice is to assign twice the cost to a diagonal alignment step, compared to a vertical or horizontal step. This effectively

Figure 6.1: Schematic illustration of an alignment path and step weights w_h , w_v , w_d for the horizontal, vertical, and diagonal steps.



prevents a bias of the alignment towards the diagonal in the case of sequences having roughly the same length. Furthermore, other alignment concepts such as partial matching, which were used in tasks like DNA alignment [118] and music harmony analysis [45], can be seen as special cases of DTW and have proven to be robust to noisy data. Moreover, Drop-DTW [39] was proposed as a differentiable alignment algorithm that can ignore sequence elements regarded as outliers.

So far, SDTW is limited to unit step weights for the horizontal, vertical, and diagonal alignment steps. As for classical DTW, varying the step weights may be beneficial in scenarios with prior knowledge about the likelihood of certain steps, or to avoid a bias towards diagonal steps for equal sequence lengths. In this chapter, we extend the SDTW algorithm to allow for arbitrary step weights. As a main contribution, we introduce efficient DP recursions for the forward and backward passes having the same computational complexity as the original SDTW approach from [30]. Furthermore, we illustrate how weighted SDTW can be parameterized towards special alignment strategies that are, e.g., robust to outliers.

6.2 Weighted SDTW

As in standard SDTW (see Chapter 4), we seek to compute a soft alignment cost and its gradient, given an input cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$. In the following section, we derive the necessary forward and backward recursions. Without loss of generality, we assume a prediction sequence of length N and a target sequence of length M .

6.2.1 Forward Algorithm

As in standard SDTW, the recursive forward algorithm computes an accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$ (see Algorithm 4.1 and [30]). Defining the horizontal (w_h), vertical (w_v), and diagonal (w_d) step weights,

Algorithm 6.1 Forward pass for weighted SDTW

```

1: Inputs:  $\mathbf{C}$ ,  $w_h$ ,  $w_v$ ,  $w_d$ ,  $\gamma$ 
2:  $\mathbf{D}(0, \cdot) = \mathbf{D}(\cdot, 0) = \infty$ 
3:  $\mathbf{D}(0, 0) = 0$ 
4: for  $m = 1, \dots, M$  do
5:   for  $n = 1, \dots, N$  do
6:     Compute  $\mathbf{D}(n, m)$  using (6.1)
7:   end for
8: end for
9: Output: SDTW cost  $\mathbf{D}(N, M) \in \mathbb{R}$ 

```

the elements of the accumulated cost matrix are recursively computed as

$$\mathbf{D}(n, m) = \mu_{\text{soft}}^{\gamma} (w_v \mathbf{C}(n, m) + \mathbf{D}(n, m-1), \quad (6.1)$$

$$w_h \mathbf{C}(n, m) + \mathbf{D}(n-1, m),$$

$$w_d \mathbf{C}(n, m) + \mathbf{D}(n-1, m-1)),$$

with the softmin function $\mu_{\text{soft}}^{\gamma}$. Note that (6.1) is a direct extension of (4.24), and we have used the distributivity property of the softmin function (see Theorem B.1) to move the local cost $\mathbf{C}(n, m)$ inside the softmin function. For $w_h = w_v = w_d = 1$, (6.1) is equal to (4.24). As in standard SDTW, the final element of the accumulated cost matrix represents the total alignment cost

$$\text{SDTW}(\mathbf{C}) = \mathbf{D}(N, M). \quad (6.2)$$

An overview of the steps for computing the SDTW forward pass, i.e., the alignment cost, is provided in Algorithm 6.1. A complete row or column of $\mathbf{D}$ is denoted by $\mathbf{D}(m, \cdot)$ and $\mathbf{D}(\cdot, n)$, respectively. Furthermore, we extend the matrix $\mathbf{D}$ by an additional column and row indexed by zero (with values set to ∞ except for $\mathbf{D}(0, 0)$ set to 0) [30]. Using the insensitivity of the softmin function to infinity inputs (see Theorem B.1), it can be shown that this definition yields the correct initialization for the recursion in (6.1).

6.2.2 Backward Algorithm

In order to train DNNs with the weighted SDTW loss function, we need to compute the gradient $\mathbf{H} := \nabla_{\mathbf{C}} \text{SDTW}(\mathbf{C}) \in \mathbb{R}^{N \times M}$ of the SDTW cost w.r.t. the cost matrix $\mathbf{C}$. While this can in principle be handled using automatic differentiation in modern deep learning frameworks, an efficient DP recursion to compute $\mathbf{H}$ is provided in [30, 94]. In this section, we extend the backward algorithm from standard SDTW (see Algorithm 4.2 and [30]) to additionally incorporate variable step weights. First, the chain rule

Algorithm 6.2 Backward pass for weighted SDTW

```

1: Inputs:  $\mathbf{C}, \mathbf{D}, w_h, w_v, w_d, \gamma$ 
2:  $\mathbf{C}(N+1, \cdot) = \mathbf{C}(\cdot, M+1) = 0$ 
3:  $\mathbf{D}(N+1, \cdot) = \mathbf{D}(\cdot, M+1) = -\infty$ 
4:  $\mathbf{D}(N+1, M+1) = \mathbf{D}(N, M)$ 
5:  $\mathbf{E}(N+1, \cdot) = \mathbf{E}(\cdot, M+1) = 0$ 
6:  $\mathbf{E}(N+1, M+1) = 1$ 
7: for  $m = M, \dots, 1$  do
8:   for  $n = N, \dots, 1$  do
9:     Compute  $\mathbf{B}^{p,q}(n, m)$  using (6.7) to (6.9)
10:    Compute  $\mathbf{E}(n, m)$  using (6.5)
11:    Compute  $\mathbf{G}(n, m)$  using (6.10)
12:    Compute  $\mathbf{H}(n, m)$  using (6.11)
13:   end for
14: end for
15: Output: gradient matrix  $\mathbf{H} \in \mathbb{R}^{N \times M}$ 

```

is applied to split the elements of the gradient

$$\mathbf{H}(n, m) := \frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} = \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)} \cdot \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)}. \quad (6.3)$$

Following the standard SDTW algorithm from [30], we again use the chain rule to split the computation of the first term of (6.3) into parts

$$\begin{aligned} \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)} &= \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m+1)} \cdot \frac{\partial \mathbf{D}(n, m+1)}{\partial \mathbf{D}(n, m)} \\ &\quad + \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n+1, m)} \cdot \frac{\partial \mathbf{D}(n+1, m)}{\partial \mathbf{D}(n, m)} \\ &\quad + \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n+1, m+1)} \cdot \frac{\partial \mathbf{D}(n+1, m+1)}{\partial \mathbf{D}(n, m)}. \end{aligned} \quad (6.4)$$

With the shorthand notations $\mathbf{E}(n, m) := \frac{\partial \mathbf{D}(N, M)}{\partial \mathbf{D}(n, m)}$ as well as $\mathbf{B}^{p,q}(n, m) := \frac{\partial \mathbf{D}(n+q, m+p)}{\partial \mathbf{D}(n, m)}$, where $p, q \in \{0, 1\}$, we can reformulate (6.4) as

$$\begin{aligned} \mathbf{E}(n, m) &= \mathbf{E}(n, m+1) \cdot \mathbf{B}^{1,0}(n, m) \\ &\quad + \mathbf{E}(n+1, m) \cdot \mathbf{B}^{0,1}(n, m) \\ &\quad + \mathbf{E}(n+1, m+1) \cdot \mathbf{B}^{1,1}(n, m). \end{aligned} \quad (6.5)$$

Using the derivative of the softmin function w.r.t. its arguments, we calculate $\mathbf{B}^{1,0}(n, m)$ as

$$\begin{aligned}\mathbf{B}^{1,0}(n, m) &= \frac{\partial \mathbf{D}(n, m+1)}{\partial \mathbf{D}(n, m)} \\ &= \frac{\exp(-(w_v \mathbf{C}(n, m+1) + \mathbf{D}(n, m)) / \gamma)}{\exp(-\mathbf{D}(n, m+1) / \gamma)}.\end{aligned}\quad (6.6)$$

Similarly, by simplifying the expression in (6.6), we obtain the solution for all combinations of $p, q \in \{0, 1\}$

$$\mathbf{B}^{1,0}(n, m) = \exp(\mathbf{D}(n, m+1) - \mathbf{D}(n, m) - w_v \mathbf{C}(n, m+1)) / \gamma, \quad (6.7)$$

$$\mathbf{B}^{0,1}(n, m) = \exp(\mathbf{D}(n+1, m) - \mathbf{D}(n, m) - w_h \mathbf{C}(n+1, m)) / \gamma, \quad (6.8)$$

$$\mathbf{B}^{1,1}(n, m) = \exp(\mathbf{D}(n+1, m+1) - \mathbf{D}(n, m) - w_d \mathbf{C}(n+1, m+1)) / \gamma. \quad (6.9)$$

Next, combining (6.5), (6.7), (6.8), and (6.9) yields the recursive computation of the matrix $\mathbf{E}$. This corresponds to the “backtracking” of traditional DTW, where the recursion is from larger to smaller indices (as opposed to the forward recursion). Similar to standard SDTW, the entries $\mathbf{E}(n, m)$ can be interpreted as the alignment probability of the elements $\mathbf{x}_n$ and $\mathbf{y}_m$ [17].

Note that in the standard SDTW formulation from [30], it holds that $\mathbf{E} = \mathbf{H}$ because $\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} = 1$ in (6.3) when using unit step weights. However, due to the modifications in (6.1), this does no longer hold, and we need to compute the entries of a correction matrix $\mathbf{G} \in \mathbb{R}^{N \times M}$ that contains the derivatives:

$$\begin{aligned}\mathbf{G}(n, m) &= \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} \\ &= \frac{\partial}{\partial \mathbf{C}(n, m)} \mu_{\text{soft}}^\gamma(w_v \mathbf{C}(n, m) + \mathbf{D}(n, m-1), w_h \mathbf{C}(n, m) + \mathbf{D}(n-1, m), w_d \mathbf{C}(n, m) + \mathbf{D}(n-1, m-1)) \\ &= \frac{w_v e^{-(w_v \mathbf{C}(n, m) + \mathbf{D}(n, m-1)) / \gamma} + w_h e^{-(w_h \mathbf{C}(n, m) + \mathbf{D}(n-1, m)) / \gamma} + w_d e^{-(w_d \mathbf{C}(n, m) + \mathbf{D}(n-1, m-1)) / \gamma}}{e^{-(w_v \mathbf{C}(n, m) + \mathbf{D}(n, m-1)) / \gamma} + e^{-(w_h \mathbf{C}(n, m) + \mathbf{D}(n-1, m)) / \gamma} + e^{-(w_d \mathbf{C}(n, m) + \mathbf{D}(n-1, m-1)) / \gamma}} \\ &= \frac{w_v e^{-(w_v \mathbf{C}(n, m) + \mathbf{D}(n, m-1)) / \gamma} + w_h e^{-(w_h \mathbf{C}(n, m) + \mathbf{D}(n-1, m)) / \gamma} + w_d e^{-(w_d \mathbf{C}(n, m) + \mathbf{D}(n-1, m-1)) / \gamma}}{e^{-\mathbf{D}(n, m) / \gamma}} \\ &= w_v \exp(-(\mathbf{D}(n, m-1) - \mathbf{D}(n, m) + w_v \mathbf{C}(n, m)) / \gamma) \\ &\quad + w_h \exp(-(\mathbf{D}(n-1, m) - \mathbf{D}(n, m) + w_h \mathbf{C}(n, m)) / \gamma) \\ &\quad + w_d \exp(-(\mathbf{D}(n-1, m-1) - \mathbf{D}(n, m) + w_d \mathbf{C}(n, m)) / \gamma).\end{aligned}\quad (6.10)$$

Finally, by inserting (6.5) and (6.10) in (6.3), we can compute the gradient of the SDTW cost w.r.t. the cost matrix as

$$\mathbf{H}(n, m) := \frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} = \mathbf{E}(n, m) \cdot \mathbf{G}(n, m). \quad (6.11)$$

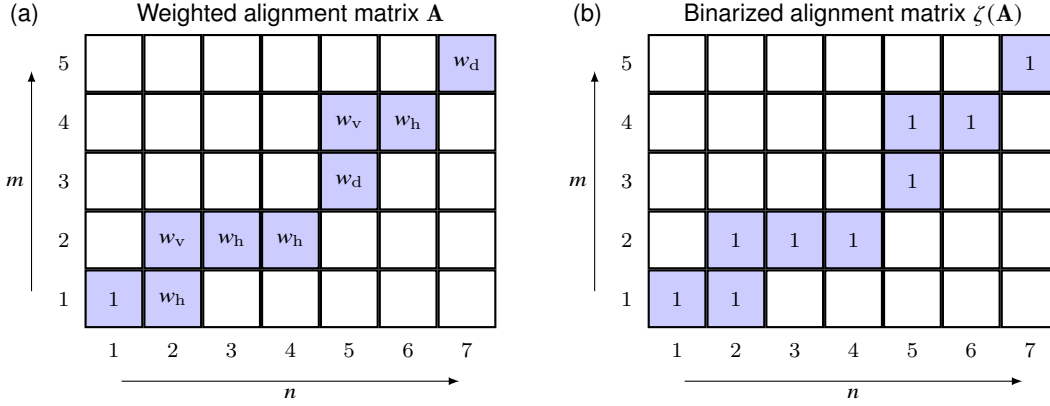


Figure 6.2: Illustration of weighted and binary alignment matrices for sequence alignment with $N = 7$ and $M = 5$. (a) Weighted alignment matrix $\mathbf{A} \in \mathbb{R}_{\geq 0}^{7 \times 5}$. (b) Binarized alignment matrix $\zeta(\mathbf{A}) \in \{0, 1\}^{7 \times 5}$. Empty cells have a value of zero.

Algorithm 6.2 gives a complete overview of the DP recursion to compute the SDTW gradient. The algorithm is based on [30]. Similar to Algorithm 6.1, we introduce an additional column and row indexed by $N + 1$ and $M + 1$, respectively, and suitably set the matrix elements to $-\infty$ and 0, which can be shown to yield the correct initialization for the backward recursion. Note that the forward and backward passes of weighted SDTW still have the algorithmic complexity $O(MN)$ of standard SDTW.

A Note on Gradients and Soft Alignments

In Chapter 4, we have formulated SDTW using binary alignment matrices $\mathbf{A} \in \{0, 1\}^{N \times M}$, where each alignment step contributes equally to the total alignment cost. In this case, the soft alignment matrix $\mathbf{E}$ was equal to the gradient matrix $\mathbf{H}$. Now, with step weights introduced, this does no longer hold. We now go beyond the DP view from Sections 6.2.1 and 6.2.2 and take on a global perspective on weighted SDTW with a focus on the soft alignment and gradient matrices.

To this end, we can define real-valued non-negative alignment matrices, i.e., $\mathbf{A} \in \mathbb{R}_{\geq 0}^{N \times M}$ (see Figure 6.2a). In this formulation, a non-zero path through $\mathbf{A}$ still encodes a valid alignment path P , while the values along the path specify the respective step weights. The total path cost is still given by the inner product $\langle \mathbf{A}, \mathbf{C} \rangle$, allowing individual steps to contribute differently to the overall cost. Using these weighted matrices, the weighted SDTW cost is given by:

$$\text{SDTW}(\mathbf{C}) = \mu_{\text{soft}}^{\gamma} \left((\langle \mathbf{A}, \mathbf{C} \rangle)_{\mathbf{A} \in \mathcal{A}} \right), \quad (6.12)$$

with $\mathcal{A}$ denoting the set of weight-encoded alignment matrices. To relate weighted and binary alignment matrices, we specify a binarization operator $\zeta : \mathbb{R}_{\geq 0}^{N \times M} \rightarrow \{0, 1\}^{N \times M}$, defined element-wise as

$$\zeta(\mathbf{A})(n, m) = \begin{cases} 1, & \text{if } \mathbf{A}(n, m) > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (6.13)$$

An illustration of a binarized alignment matrix $\zeta(\mathbf{A})$ is provided in Figure 6.2b.

Recall that the SDTW gradient with respect to the local cost matrix $\mathbf{C}$ is given by

$$\mathbf{H} = \sum_{\mathbf{A} \in \mathcal{A}} p(\mathbf{A} \mid \mathbf{C}) \mathbf{A}, \quad (6.14)$$

where $\mathcal{A}$ denotes the set of valid alignment matrices and $p(\mathbf{A} \mid \mathbf{C})$ the Gibbs distribution over alignments induced by SDTW (see Section 4.3). We interpret $\mathbf{H} \in \mathbb{R}_{\geq 0}^{N \times M}$ as a *gradient matrix*: each entry $\mathbf{H}(n, m)$ quantifies how strongly the local cost $\mathbf{C}(n, m)$ influences the final alignment cost. In other words, $\mathbf{H}$ represents the expected weighted alignment under the soft path distribution.

In addition, we now view the *soft alignment matrix* $\mathbf{E} \in [0, 1]^{N \times M}$ as

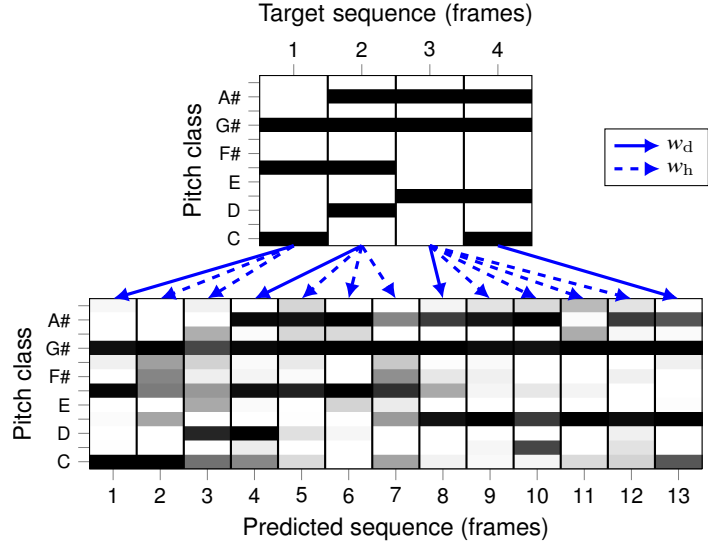
$$\mathbf{E} = \sum_{\mathbf{A} \in \mathcal{A}} p(\mathbf{A} \mid \mathbf{C}) \zeta(\mathbf{A}), \quad (6.15)$$

where ζ removes step weights and only retains path structure. The entries $\mathbf{E}(n, m) \in [0, 1]$ can be interpreted as the probability that an alignment path passes through cell (n, m) . Thus, while $\mathbf{H}$ reflects weighted influence on the loss, $\mathbf{E}$ captures the underlying soft alignment. Note that in the case of equal step weights, as considered in Chapter 4, all valid alignment matrices are binary by definition. In this setting, the binarization operator ζ reduces to the identity, and the soft alignment matrix is equal to the gradient matrix, that is, $\mathbf{E} = \mathbf{H}$.

6.3 Experiments and Evaluation

In this section, we consider the task of Pitch Class Estimation (PCE) as a controlled test scenario to indicate the potential of SDTW with variable step weights. Pitch class features are widely used in MIR for, e.g., music synchronization or harmony analysis [98]. Compared to Multi-Pitch Estimation (MPE) with typically high-dimensional output features, the twelve-dimensional pitch class vectors of PCE allow for an easy visualization and performance analysis of the estimation algorithms. In the following, we first introduce the setup for our PCE experiment and then provide insights on DNN training behavior under the weighted SDTW loss as well as the underlying SDTW alignments for different horizontal step weights.

Figure 6.3: Frame-wise alignment between weakly aligned pitch class target vectors and DNN predictions. Diagonal and horizontal alignment steps are illustrated in solid and dashed lines, respectively.



6.3.1 Experimental Setup

We train and evaluate on the SWD [153] which, for every song, contains audio recordings (piano and singer) and strongly aligned pitch class annotations. Following [146, 154], we use a musically motivated CNN architecture with less than 50000 trainable parameters to predict N pitch class activation vectors $\mathbf{x}_n \in \mathbb{R}^{12}$ (see Figure 6.3, bottom) for $N + \text{context}$ input audio frames. The input audio is transformed using a HCQT [14]. For training, we use weakly aligned target sequences consisting of binary vectors $\mathbf{y}_m \in \mathbb{R}^{12}$, in which consecutive vectors differ (see Figure 6.3, top). We refer to Section 5.3 and [167] for details on the network architecture and training procedure. An implementation of the weighted SDTW loss using the cuda and pytorch frameworks is available at <https://github.com/groupmm/weightedSDTW>.

6.3.2 Interpretation of Alignment Step Directions

In our experiments, we represent the indices of the DNN predictions n and the target vectors m on the horizontal and vertical axis of the SDTW cost matrix, respectively. That is, a horizontal alignment step corresponds to a repetition of the same target vector for subsequent predictions, a vertical step corresponds to the alignment of two target vectors to the same predicted frame, and a diagonal step advances both the target and the predicted frame (see Figure 6.3 for an illustration of diagonal and horizontal steps).

6.3.3 Quantitative Analysis of the F-measure

In the following experiments, we fix the vertical and diagonal step weights $w_v = w_d = 1$ and only vary the horizontal step weight $w_h \in \{0.0, 0.2, 0.6, 1.0\}$ when using the weighted SDTW loss function.

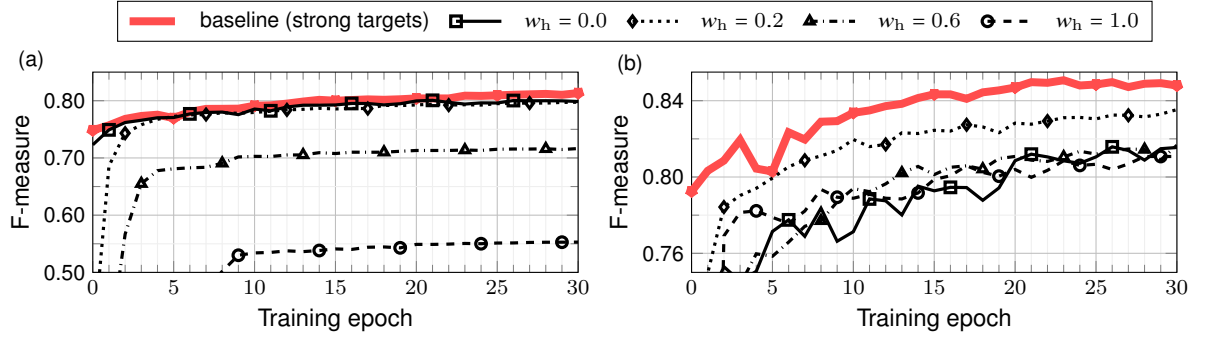


Figure 6.4: (a) Mean F-measure of ten randomly initialized training runs and (b) F-measure of best training run.

Furthermore, we train a baseline network on strong targets with an element-wise MSE loss. We repeat every training ten times using different random initializations.

In Figure 6.4a, we visualize the mean F-measure over all training runs. While training with a horizontal step weight $w_h = 1$ results in training instabilities due to degenerated alignments in the early training stage [167] (see also Section 6.3.4 for a discussion of alignment degenerations) and thus a low F-measure, reducing w_h gradually improves the F-measure. For very low values $w_h \rightarrow 0$, training runs converge fast and the prediction F-measure is almost on par with the baseline trained on strongly aligned targets.

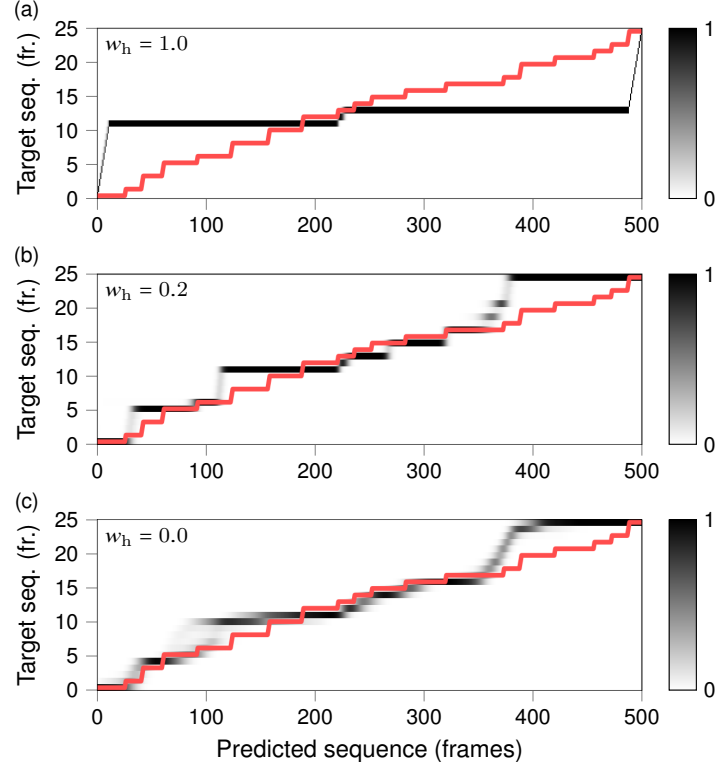
Analyzing the F-measure of the best training run for each setting in Figure 6.4b, we find that the settings with $w_h \in \{0, 0.6, 1\}$ perform almost equal. However, a very small horizontal step weight $w_h = 0.2$ significantly improves the F-measure. Thus, we argue that carefully adjusting the step weights is beneficial compared to using the trivial choices of step weights 1 or 0, while improving convergence properties of DNN training.

6.3.4 Alignment Degeneration and Robustness to Outliers

In the following section, we qualitatively discuss the effects of varying the horizontal step weight w_h by visualizing the alignment matrix $\mathbf{E}$ after the first training epoch (where the predictions are still of low quality), see Figure 6.5.

When using a high audio frame rate, the number of audio frames is typically much larger than the number of note event changes (i.e., $N \gg M$). Thus, each target vector is usually repeated many times in an alignment. As all repetitions of a target vector contribute equally to the total cost in the case of equal step weights, e.g., $w_h = w_d = w_v = 1$, poor PCE vector predictions significantly influence the estimated alignment. In the case of very noisy predictions—as in our example—the alignment is dominated by few target vectors which “on average” minimize the total alignment cost (see Figure 6.5a, where an alignment is largely degenerated to only two target vectors). Obviously, a mismatch between the SDTW alignment and the reference alignment may yield poor parameter updates in DNN training, explaining the low F-measure discussed in Section 6.3.3.

Figure 6.5: SDTW soft alignments $\mathbf{E}$ for different horizontal step weights. (a) $w_h = 1.0$. (b) $w_h = 0.2$. (c) $w_h = 0.0$. The reference alignment is illustrated in red.



To account for the repetition of target vectors in the considered PCE scenario, we investigate a reduction of the horizontal step weight w_h . In the limit case $w_h = 0$, the total SDTW cost contains exactly one alignment for every target frame (coming from a vertical or diagonal step), while all following repetitions of the same target frame (horizontal steps) come at no additional cost. That is, after a single correct prediction of a target vector, the alignment is not impaired by subsequent noisy predictions of the same target. Analyzing the alignment matrix $\mathbf{E}$ for $w_h = 0.2$ and $w_h = 0$ in Figure 6.5b and Figure 6.5c, respectively, the alignments are indeed much closer to the reference and thus allow for a much more stable DNN training.

The robustness towards outliers by ignoring repeated elements in the computation of the total alignment cost can be seen as a parameterization of SDTW towards other algorithms such as partial matching [45, 118] or Drop-DTW [39], which also allow for skipping potentially noisy sequence elements. However, weighted SDTW is not restricted to this particular scenario and can be used to, e.g., eliminate the bias towards the diagonal in the case of sequences with equal lengths using parameters $\{w_h = w_v = 1, w_d = 2\}$.

6.4 Further Notes

In this chapter, we extended the SDTW algorithm to allow for variable step weights. As our main technical contribution, we derived efficient DP recursions for the computation of the alignment cost as well as the

gradient. We used the weighted SDTW loss for the task of DNN-based PCE, where we investigated the effect when reducing the horizontal step weight. In particular, we showed that a suitable weighting scheme may lead to a more stable DNN training in the case of noisy predictions. Furthermore, weighted SDTW can be used to eliminate the bias towards the diagonal of classic SDTW in tasks with equal sequence lengths.

7 Subsequence Variants of SDTW

This chapter is based on [166]. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

So far, we have assumed in classical DTW and SDTW formulations that the compared sequences correspond precisely at their start and end points. This assumption is often violated in practice when weakly aligned data suffers from boundary mismatches. In classical DTW, this issue is addressed by Subsequence DTW (subDTW), which allows one sequence to match a subsequence of the other under preserved monotonicity but relaxed boundary conditions.

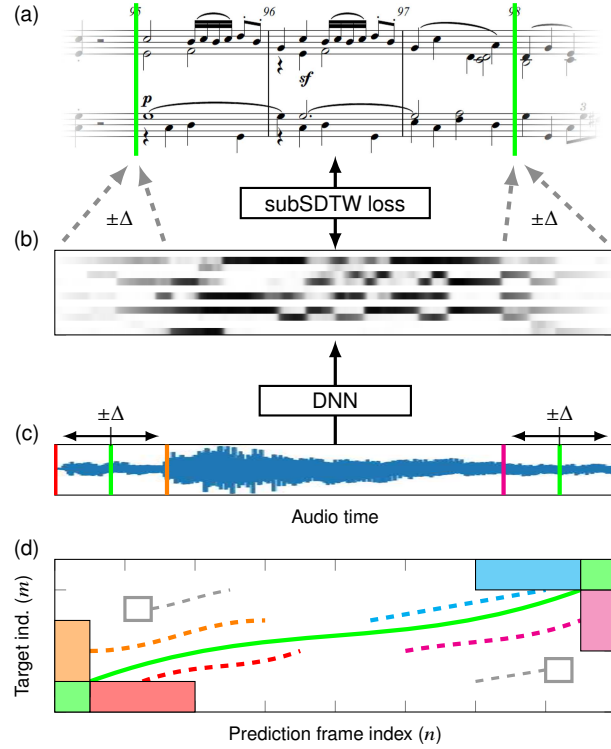
This chapter transfers this idea to differentiable alignment frameworks and introduces Subsequence SDTW (subSDTW), preserving differentiability and efficient DP implementations for both loss and gradient computation. Controlled experiments demonstrate improved robustness when training from weakly aligned input–target pairs with imprecise temporal boundaries, thereby removing the restrictive structural boundary requirements of standard SDTW.

The remainder of this chapter is structured as follows. In Section 7.1, we motivate the need for differentiable alignment methods being able of handling imprecise boundary annotations. In Section 7.2 we extend the differentiable SDTW framework to support flexible boundaries and introduce subSDTW, the differentiable counterpart to subDTW. We derive efficient DP recursions for both loss and gradient computation, enabling end-to-end training from weakly aligned subsequences. We evaluate the subSDTW algorithm in Section 7.3 in a controlled experiment using weakly aligned score–audio pairs with boundary mismatches as training data for MPE. We conclude in Section 7.4 and give an outlook to future work.

7.1 Motivation for Subsequence Alignment

Training DNNs on large sequential datasets is central to tasks in ASR [58], motion alignment [25], and music transcription [12], typically relying on frame-level aligned input–target pairs. However, such

Figure 7.1: Schematic overview of SDTW alignment with boundary mismatch. (a) Musical score, from which the weak targets are derived. (b) DNN predictions. (c) Input audio with boundary uncertainty Δ . (d) subSDTW cost matrix with subsequence boundary conditions and corresponding alignment paths.



strongly aligned data is often costly to obtain. In contrast, weakly aligned segments, where only start and end boundaries are known, are more accessible. In ASR, these targets may consist of unaligned text or phoneme sequences; in MIR, this could be weakly aligned note sequences derived from a musical score (see Figure 7.1a).

Although SDTW supports learning from weak alignments, it still requires exact boundary correspondences. In practice, such as with automatically paired audio–score segments, precise boundaries are rarely available and hard to annotate (see Figure 7.1c). Misalignments at the boundaries violate the assumptions of these losses, limiting training effectiveness. To address this *subsequence* alignment problem, classical DTW has been extended with flexible boundary conditions [22, 98], enabling alignment in scenarios where, e.g., the first sequence corresponds only to a subsequence of the second. However, the non-differentiable DTW algorithm cannot be applied in end-to-end DNN training. In the following section, we extend the differentiable SDTW framework to support flexible boundaries and introduce Subsequence SDTW (subSDTW), the differentiable counterpart to subDTW.

7.2 Subsequence SDTW

In this section, we describe the computation of the subSDTW loss, closely following prior work [30, 94, 168], and extending the algorithm with flexible boundary conditions. The loss computation is based on a cost

matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ that is computed from two sequences $X = (x_1, \dots, x_N)$ and $Y = (y_1, \dots, y_M)$ [30]. The set $\mathcal{I}$ of all cells in the cost matrix is given by $\mathcal{I} := [1 : N] \times [1 : M]$.

7.2.1 Boundary Conditions and Step Weights

To model boundary mismatches in the subSDTW framework, we define start and end boundary conditions $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$ as sets of cells. All alignment paths must start in one of the start boundary cells, and end in one of the end boundary cells. To compensate for a low alignment cost due to shorter paths, we introduce associated weights $w_{\text{start}}^{(n,m)} \in \mathbb{R}$ for $(n, m) \in \mathcal{B}_{\text{start}}$ and $w_{\text{end}}^{(n,m)} \in \mathbb{R}$ for $(n, m) \in \mathcal{B}_{\text{end}}$ for the cells in the start and end boundary condition set, respectively. In this chapter, we parameterize these weights as multiples of the local cost value:

$$w_{\text{start}}^{(n,m)} = \tilde{w}_{\text{start}}^{(n,m)} \cdot \mathbf{C}(n, m) \quad (7.1)$$

$$w_{\text{end}}^{(n,m)} = \tilde{w}_{\text{end}}^{(n,m)} \cdot \mathbf{C}(n, m), \quad (7.2)$$

with scaling factors $\tilde{w}_{\text{start}}^{(n,m)}, \tilde{w}_{\text{end}}^{(n,m)} \in \mathbb{R}$. An example parameterization of these factors is provided in Section 7.3.2, while a more general discussion of arbitrary start and end weights can be found in Chapter 10. While the experiment illustrated in Figure 7.3 and described in Section 7.3 assigns boundary cells only at the edges of the cost matrix, the subSDTW algorithm allows for arbitrary boundary conditions (see Figure 7.1d). Furthermore, following [94, 168], we define step sizes $\mathcal{S} = \{(i_s, j_s) \mid s \in [1 : S]\}$ as a set of integer pairs with monotonicity constraint $i_s \geq 0, j_s \geq 0, i_s + j_s \geq 1$ for all $(i_s, j_s) \in \mathcal{S}$, and step weights $w_{\text{step}}^{(i_s, j_s)} \in \mathbb{R}$ corresponding to the step sizes $(i_s, j_s) \in \mathcal{S}$.

7.2.2 Forward Recursion

Our goal is to iteratively compute an accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$ with an efficient DP recursion [30]. To this end, for every cell $(n, m) \in \mathcal{I}$ we define a vector of accumulated incoming cost $\mathbf{f}^{(n,m)} \in \mathbb{R}^{S+1}$, with elements $\mathbf{f}_s^{(n,m)}$ for $[1 : S]$ used for the cost of incoming steps, and element $\mathbf{f}_{S+1}^{(n,m)}$ used for cost from a start boundary cell. Identical to weighted SDTW introduced in Chapter 6,

$$\mathbf{f}_s^{(n,m)} := \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I}, \\ \mathbf{D}(n - i_s, m - j_s) + w_{\text{step}}^{(i_s, j_s)} \mathbf{C}(n, m), & \text{else} \end{cases} \quad (7.3)$$

denotes the cost of incoming steps (i_s, j_s) with $s \in [1 : S]$. The additional contribution in case (n, m) is in the start boundary conditions is given by

$$\mathbf{f}_{S+1}^{(n,m)} := \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}}, \\ \tilde{w}_{\text{start}}^{(n,m)} \cdot \mathbf{C}(n, m), & \text{else,} \end{cases} \quad (7.4)$$

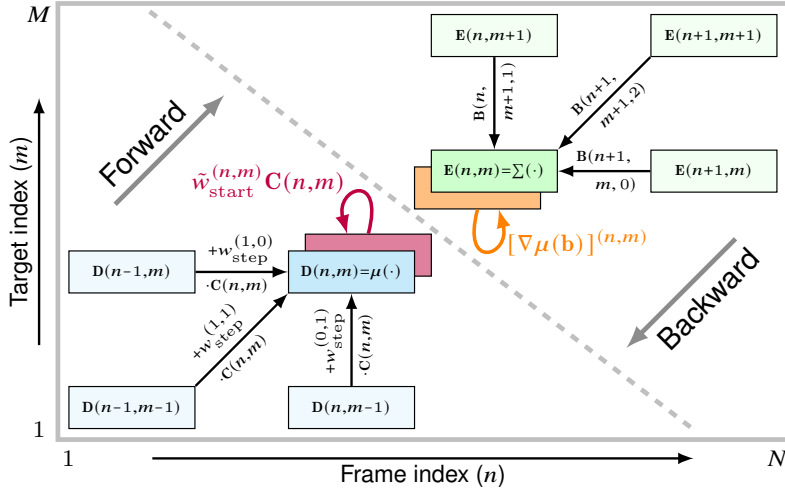


Figure 7.2: Schematic overview of subSDTW forward and backward passes for step sizes $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$. In the forward pass (blue), each cell $\mathbf{D}(n, m)$ is updated based on previous cells $\mathbf{D}(n - i_s, m - j_s)$ and (optionally) a start boundary condition (purple). In the backward pass (green), each cell $\mathbf{E}(n, m)$ is updated based on previous cells $\mathbf{E}(n + i_s, m + j_s)$ and (optionally) an end boundary condition (orange). Except for the update at boundary cells (purple and orange), the subSDTW algorithm is identical to SDTW.

where $\tilde{w}_{\text{start}}^{(n,m)}$ can be used to increase the current path cost at start boundary cells to prevent alignment collapse to degenerate paths. As in standard SDTW [30], we compute the soft minimum over cost values from incoming steps and specify the forward recursion:

$$\mathbf{D}(n, m) = \mu_{\text{soft}}^{\gamma} \left(\mathbf{f}^{(n,m)} \right). \quad (7.5)$$

The forward update of $\mathbf{D}(n, m)$ is illustrated in Figure 7.2. For later use in the backward recursion, we save the gradient of the updated local cost $\mathbf{D}(n, m)$ w.r.t. the components of $\mathbf{f}^{(n,m)}$ in a backtracking tensor $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ with entries

$$\mathbf{B}(n, m) := \nabla \mu_{\text{soft}}^{\gamma} \left(\mathbf{f}^{(n,m)} \right) \in \mathbb{R}^{S+1}, \quad (7.6)$$

where $\mathbf{B}(n, m, s)$ for $s \in [1 : S]$ corresponds to the cost contributions from incoming steps, and $\mathbf{B}(n, m, S+1)$ corresponds to contribution from the start boundary condition. After updating all cells $(n, m) \in \mathcal{I}$, we collect all accumulated cost values at cells given by the end boundary condition, including an additional term $\tilde{w}_{\text{end}}^{(n,m)} \cdot \mathbf{C}(n, m)$ to compensate for short paths, in a collection $\mathbf{b} := (b^{(n,m)})_{(n,m) \in \mathcal{B}_{\text{end}}}$ with elements

$$b^{(n,m)} := \mathbf{D}(n, m) + \tilde{w}_{\text{end}}^{(n,m)} \cdot \mathbf{C}(n, m). \quad (7.7)$$

The total cost SDTW ($\mathbf{C}$) $\in \mathbb{R}$ is given by evaluating the softmin function μ over the cost at the end boundary cells specified by $\mathbf{b}$:

$$\text{SDTW}(\mathbf{C}) = \mu_{\text{soft}}^{\gamma} (\mathbf{b}). \quad (7.8)$$

Algorithm 7.1 subSDTW forward pass

```

1: Inputs:
2: Cost matrix  $\mathbf{C} \in \mathbb{R}^{N \times M}$  indexed by cells  $\mathcal{I}$ 
3: Boundary conditions  $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$ 
4: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
5: Start, end, and step weights  $\tilde{w}_{\text{start}}^{(n,m)}, \tilde{w}_{\text{end}}^{(n,m)}, w_{\text{step}}^{(i_s, j_s)}$ 
6: for  $m = 1, \dots, M$  do
7:   for  $n = 1, \dots, N$  do
8:     for  $s = 1, \dots, S$  do
9:        $\mathbf{f}_s^{(n,m)} = \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I}, \\ \mathbf{D}(n - i_s, m - j_s) + w_{\text{step}}^{(i_s, j_s)} \mathbf{C}(n, m), & \text{else,} \end{cases}$ 
10:    end for
11:     $\mathbf{f}_{S+1}^{(n,m)} = \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}}, \\ \tilde{w}_{\text{start}}^{(n,m)} \mathbf{C}(n, m), & \text{else.} \end{cases}$ 
12:     $\mathbf{D}(n, m) = \mu(\mathbf{f}^{(n,m)})$ 
13:    Save  $\mathbf{B}(n, m) = \nabla \mu(\mathbf{f}^{(n,m)})$  for backward
14:  end for
15: end for
16:  $\mathbf{b} = (\mathbf{D}(n, m) + \tilde{w}_{\text{end}}^{(n,m)} \mathbf{C}(n, m))_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
17: Output: subSDTW cost  $\text{SDTW}(\mathbf{C}) = \mu(\mathbf{b}) \in \mathbb{R}$ 

```

In comparison, for standard SDTW [30], the total cost would be given by the final cell of the accumulated cost matrix $\mathbf{D}(N, M)$. The subSDTW forward algorithm is summarized in Algorithm 7.1.

7.2.3 Backward Recursion

When training DNNs, we need to obtain the gradient $\mathbf{H} \in \mathbb{R}^{N \times M}$ of the total cost $\text{SDTW}(\mathbf{C})$ with respect to the local cost matrix $\mathbf{C}$, with elements defined as $\mathbf{H}(n, m) := \partial \text{SDTW}(\mathbf{C}) / \partial \mathbf{C}(n, m)$. Following [30, 168], we describe the gradient calculation and highlight the differences between standard SDTW and subSDTW. First, we apply the chain rule to separate the contribution of cost $\mathbf{C}(n, m)$ to the accumulated cost $\mathbf{D}(n, m)$ (denoted by $\mathbf{G}(n, m)$), and the contribution of accumulated cost $\mathbf{D}(n, m)$ to the total cost (denoted by $\mathbf{E}(n, m)$) [30, 94]:

$$\mathbf{H}(n, m) = \underbrace{\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{D}(n, m)}}_{:= \mathbf{E}(n, m)} \cdot \underbrace{\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)}}_{:= \mathbf{G}(n, m)} \quad (7.9)$$

with $\mathbf{E}, \mathbf{G} \in \mathbb{R}^{N \times M}$. The value of $\mathbf{E}(n, m)$ quantifies how much the accumulated cost $\mathbf{D}(n, m)$ affects the total cost. In subSDTW, the value of $\mathbf{D}(n, m)$ can contribute to the total cost in two different ways: by influencing following cells in the accumulated cost matrix [30], and by contributing to the total cost in

case (n, m) is contained in the set of end boundary cells. We therefore write

$$\mathbf{E}(n, m) = \sum_{s=1}^S \frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} \cdot \frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} + [\nabla \mu(\mathbf{b})]^{(n, m)}, \quad (7.10)$$

where the first term in (7.10) denotes influence of a following cell on the total cost and, as in standard SDTW [30], is given by previously computed entries of the $\mathbf{E}$ -matrix:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{E}(n + i_s, m + j_s), & \text{else.} \end{cases} \quad (7.11)$$

The second term in (7.10), also found in standard SDTW [30], denotes influence of the current cell on a following cell and is given by the previously computed backtracking matrix:

$$\frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{B}(n + i_s, m + j_s, s), & \text{else.} \end{cases} \quad (7.12)$$

The third term in (7.10) is introduced for subSDTW and denotes potential contribution coming from the end boundary conditions:

$$[\nabla \mu(\mathbf{b})]^{(n, m)} := \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n, m)}, & \text{if } (n, m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases} \quad (7.13)$$

The update of $\mathbf{E}$ in the backward pass is illustrated in Figure 7.2. The contribution of the local cost $\mathbf{C}(n, m)$ on the accumulated cost $\mathbf{D}(n, m)$ is given by the chain rule as follows:

$$\mathbf{G}(n, m) = \sum_{s=1}^{S+1} \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{f}_s^{(n, m)}} \cdot \frac{\partial \mathbf{f}_s^{(n, m)}}{\partial \mathbf{C}(n, m)}, \quad (7.14)$$

where the first term in (7.14) is given by the backtracking matrix:

$$\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{f}_s^{(n, m)}} = \mathbf{B}(n, m, s), \quad (7.15)$$

and the second term in (7.14) is given by either the step weight (as in weighted SDTW [168]), or, for subSDTW in the case (n, m) is in the start boundary conditions, the respective boundary weight:

$$\frac{\partial \mathbf{f}_s^{(n, m)}}{\partial \mathbf{C}(n, m)} = \begin{cases} w_{\text{step}}^{(i_s, j_s)}, & \text{if } s \in [1 : S], \\ \tilde{w}_{\text{start}}^{(n, m)}, & \text{if } s=S+1 \text{ and } (n, m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases} \quad (7.16)$$

Algorithm 7.2 subSDTW backward pass

```

1: Inputs:
2: Backtracking tensor  $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ 
3: Start and end boundary conditions  $\mathcal{B}_{\text{end}}, \mathcal{B}_{\text{start}} \subseteq \mathcal{I}$ 
4: End boundary costs  $\mathbf{b} := (b^{(n,m)})_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
5: Step and start weights  $w_{\text{step}}^{(i_s, j_s)}, \tilde{w}_{\text{start}}^{(n,m)}$ 
6: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
7: for  $m = M, \dots, 1$  do
8:   for  $n = N, \dots, 1$  do
9:     for  $s = 1 \dots S$  do
10:       $\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{E}(n+i_s, m+j_s), & \text{else.} \end{cases}$ 
11:       $\frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n, m)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{B}(n+i_s, m+j_s, s), & \text{else.} \end{cases}$ 
12:    end for
13:     $[\nabla \mu(\mathbf{b})]^{(n,m)} = \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n,m)}, & \text{if } (n, m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases}$ 
14:     $\mathbf{E}(n, m) = \sum_{s=1}^S \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} \cdot \frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n, m)} + [\nabla \mu(\mathbf{b})]^{(n,m)}$ 
15:    for  $s=1, \dots, S+1$  do
16:       $\frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n, m)} = \begin{cases} w_{\text{step}}^{(i_s, j_s)}, & \text{if } s \in [1 : S], \\ \tilde{w}_{\text{start}}^{(n,m)}, & \text{if } s=S+1 \text{ and } (n, m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases}$ 
17:    end for
18:     $\mathbf{G}(n, m) = \sum_{s=1}^{S+1} \mathbf{B}(n, m, s) \cdot \frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n, m)}$ 
19:     $\mathbf{H}(n, m) = \mathbf{E}(n, m) \cdot \mathbf{G}(n, m)$ 
20:  end for
21: end for
22: Output: Gradient  $\mathbf{H} \in \mathbb{R}^{N \times M}$ 

```

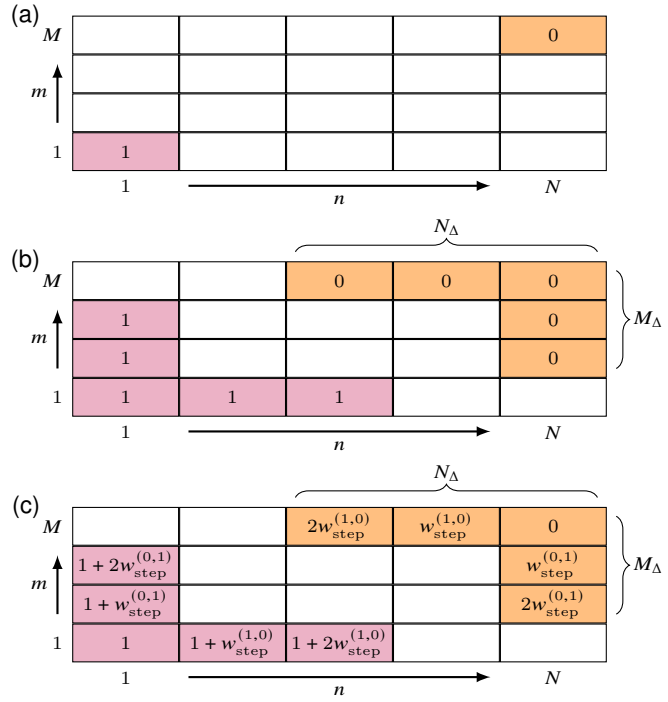
We summarize the complete backward recursion in Algorithm 7.2

7.3 Experiments

In this section, we evaluate subSDTW in the context of DNN training. As an illustrative example from a relevant MIR task, we choose Multi-Pitch Estimation (MPE) as our case study. Our primary objective is not to advance the state of the art in MPE, but rather to analyze the impact of weak alignment and boundary mismatch in a controlled training setup. The full code for the experiments, including a CUDA-compatible implementation of the subSDTW loss, is available online.¹¹

¹¹ <https://github.com/groupmm/subsequenceSDTW>.

Figure 7.3: Start (purple) and end (orange) boundary conditions with corresponding weights for different training configurations. (a) SDTW. (b) subSDTW. (c) subSDTW-W.



7.3.1 Experimental Setup

We adopt a simple yet representative MPE architecture: a single convolutional stack based on the Python implementation of the *Onsets and Frames* model [63], which processes Mel spectrograms to predict frame-wise pitch activity. The model is pre-trained on the MAESTRO dataset [64], which provides piano recordings with strongly aligned MIDI annotations from a Disklavier.

For fine-tuning, we use weakly aligned data from the Beethoven Piano Sonata Dataset (BPSD) (see Chapter 3 and [170]), containing diverse recordings of Beethoven’s piano sonatas. Score–audio pairs are created by matching four-measure pitch label segments (from the score) with corresponding audio. Weak targets Y consist of non-repeating, simultaneously active notes derived from these segments. To evaluate subSDTW, we simulate boundary mismatches by randomly shifting start and end points of audio segments within $[-\Delta, \Delta]$.

Fine-tuning is performed on batches of size 8 using the subSDTW loss with $\gamma = 0.1$, standard step sizes $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$, and step weights $\mathcal{W}_{\text{step}} = \{0.1, 1.0, 1.0\}$. The reduced weight for the $(1, 0)$ step improves training stability, see Chapter 6 and [168]. We use BCE as the local cost and optimize for 5,000 steps with Adam [74]. For architectural and loss function details, see Chapter 6, [63, 168] and our code.¹¹

7.3.2 Evaluation

We evaluate several loss configurations with increasing flexibility in handling boundary mismatch. Results on a held-out test set from the BPSD for frame-wise precision, recall, and F-measure are found in Table 7.1.

Pretrained Model (Pretrained). As a first baseline, we take the model that was pretrained on MAESTRO, evaluate it on the BPSD test set, and obtain a rather low F-measure of 0.58.

Strong targets (Strong). Next, we fine-tune the pretrained model with strongly-aligned reference targets from the BPSD, using an element-wise BCE loss. The resulting F-measure of 0.67, which is relatively low due to the complex acoustic conditions within the BPSD, can be seen as an upper bound for model performance.

Standard SDTW (SDTW). As the first configuration with weakly aligned targets, we use standard SDTW, which assumes perfect alignment at the sequence boundaries. To replicate standard SDTW within the subSDTW framework, we define singular start and end points with $\mathcal{B}_{\text{start}} = \{(1, 1)\}$, $\mathcal{B}_{\text{end}} = \{(N, M)\}$, and assign default boundary weights $\tilde{w}_{\text{start}}^{(1,1)} = 1$ and $\tilde{w}_{\text{end}}^{(N,M)} = 0$, as illustrated in Figure 7.3a. We increase the range of segment boundary mismatch from ± 0.0 s (no mismatch) up to ± 2.0 s and observe a slight decrease in F-measure from 0.67 to 0.63. Although SDTW cannot model boundary mismatches, performance is relatively stable even for large Δ . We explain this with the fact that we randomly sample the boundary mismatch from the interval $[-\Delta, \Delta]$, and the “no mismatch” assumption of SDTW is, on average, a good strategy for this scenario.

Subsequence SDTW without boundary weights (subSDTW). To allow for boundary mismatches, we extend the set of admissible boundary conditions. For this configuration, illustrated in Figure 7.3b, we define

$$\begin{aligned}\mathcal{B}_{\text{start}} &= \{(1, 1), (2, 1), \dots, (N_\Delta, 1), (1, 2), \dots, (1, M_\Delta)\} \\ \mathcal{B}_{\text{end}} &= \{(N, M), (N-1, M), \dots, (N-N_\Delta, M), (N, M-1), \dots, (N, M-M_\Delta)\}\end{aligned}\quad (7.17)$$

where N_Δ and M_Δ describe the maximum number of left-out prediction or weak target frames, determined by the maximum allowed shift Δ of the audio boundaries. All boundary weights are kept uniform $\tilde{w}_{\text{start}}^{(n,m)} = 1$, $\tilde{w}_{\text{end}}^{(n,m)} = 0$ for all $(n, m) \in \mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}}$ as to not penalize short alignment paths resulting from extended boundary conditions. Analyzing the prediction performance in Table 7.1, we find a severe decrease in F-measure from 0.65 for $\Delta = 0.5$ s to 0.41 for $\Delta = 2.0$ s. Under this parameterization, the subSDTW loss does not compute a meaningful alignment, since it collapses to the shortest possible alignment path under the allowed boundary conditions, without any penalty for short paths.

Subsequence SDTW with cell-dependent boundary weights (subSDTW-W). We enhance the previous setup by assigning cell-dependent boundary weights that reflect the difference in accumulated cost when starting from or ending in each boundary cell. Using the same sets $\mathcal{B}_{\text{start}}$ and $\mathcal{B}_{\text{end}}$ as SDTW (see (7.17)),

Table 7.1: Frame-wise precision, recall, and F-measure for different training configurations.

Configuration	Δ	Precision	Recall	F-measure
Pretrained	-	0.60	0.61	0.58
Strong	-	0.70	0.65	0.67
SDTW	0.0 s	0.70	0.65	0.67
	0.5 s	0.70	0.64	0.66
	1.0 s	0.70	0.61	0.65
	2.0 s	0.72	0.57	0.63
subSDTW	0.5 s	0.69	0.63	0.65
	1.0 s	0.73	0.53	0.61
	2.0 s	0.77	0.28	0.41
subSDTW-W	0.5 s	0.69	0.66	0.67
	1.0 s	0.69	0.66	0.67
	2.0 s	0.70	0.63	0.66

we define the weights as

$$\tilde{w}_{\text{start}}^{(n,m)} = \begin{cases} 1 + (n-1) \cdot w_{\text{step}}^{(1,0)}, & \text{if } m = 1, \\ 1 + (m-1) \cdot w_{\text{step}}^{(0,1)}, & \text{if } n = 1, \end{cases} \quad (7.18)$$

$$\tilde{w}_{\text{end}}^{(n,m)} = \begin{cases} (N-n) \cdot w_{\text{step}}^{(1,0)}, & \text{if } m = M, \\ (M-m) \cdot w_{\text{step}}^{(0,1)}, & \text{if } n = N, \end{cases} \quad (7.19)$$

where $w_{\text{step}}^{(1,0)}$ and $w_{\text{step}}^{(0,1)}$ are the corresponding step weights. This configuration is illustrated in Figure 7.3c and exhibits in Table 7.1 stable and high scores for the F-measure. Even for high $\Delta = \pm 2$ s, the F-measure of 0.66 is only slightly below the reference trained with strongly aligned targets, demonstrating the effectiveness of the proposed subSDTW loss and a task-specific parameterization of the boundary weights.

7.4 Further Notes

In this chapter, we proposed the subSDTW algorithm, an extension of subDTW and classical SDTW that incorporates flexible boundary conditions in a differentiable alignment framework. We derived efficient DP recursions for both the loss and gradient computations, and demonstrated that a suitably parameterized subSDTW loss enables stable DNN training from weakly aligned targets, even in the presence of temporal mismatches at segment boundaries. In future work, subSDTW can be applied to a broad range of scenarios involving in-the-wild data with unknown or imprecise annotation quality. An additional promising direction is the extension of the SDTW loss to other alignment paradigms such as the Smith-Waterman algorithm [134].

8 Understanding SDTW: Target Artifacts and Prediction Quality

This chapter is based on [164]. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

Even when training is stable, SDTW can cause certain artifacts in the predictions, for example, temporal blurring. Analyzing the causes of these effects is difficult, since the internal alignment is hard to interpret.

This chapter reformulates the SDTW gradient as an equivalent representation with strongly-aligned modified targets, which are analytically tractable and easy to interpret. We analyze how certain SDTW hyperparameters influence prediction quality and training dynamics, using an MPE scenario as a case study. This reformulation provides an interpretable view of the SDTW training behavior and clarifies how the alignment-induced target modifications affect the learning outcome.

The remainder of this chapter is structured as follows. Section 8.1 introduces the problem and methodology. In Section 8.2.1, we reformulate the gradient of the SDTW loss into the canonical form of element-wise MSE and BCE losses, yielding the so-called modified targets. Section 8.2.2 discusses the properties of these modified targets. Sections 8.3.1 and 8.3.2 outline our experimental framework and demonstrate the impact of different training configurations using SDTW. Finally, we discuss our findings and their implications in Section 8.3.3 and conclude in Section 8.4.

8.1 Problem Statement and Observed Artifacts

We consider the task of training a DNN that takes an input sequence and predicts features $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ with $\mathbf{x}_n \in \mathbb{R}^D$, and frame index $n \in \{1, 2, \dots, N\}$. For example, in the case of MPE, the input sequence can be a spectral representation of an audio recording and X is a sequence of estimated pitch vectors. Ideally, to train such a network, we have access to a strongly aligned target sequence $Y^{\text{strong}} = (\mathbf{y}_1^{\text{strong}}, \dots, \mathbf{y}_N^{\text{strong}})$

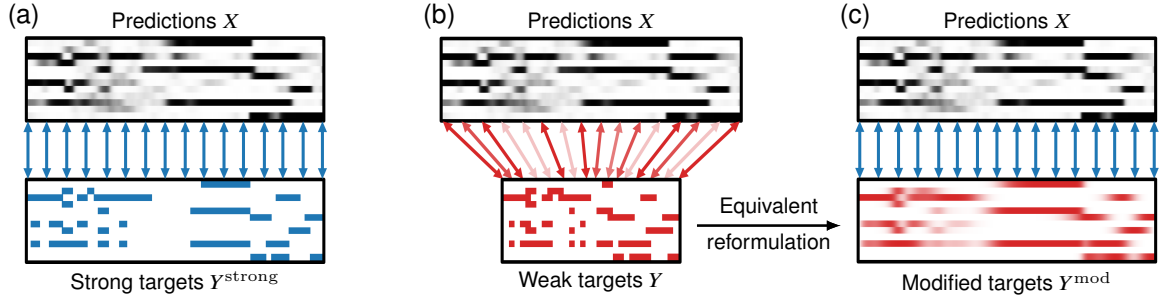


Figure 8.1: Overview of different training and alignment strategies. (a) Strong targets Y^{strong} with direct frame-wise correspondence to the predictions X . (b) Weak targets Y that require alignment to the predictions. (c) Reformulation of weakly aligned targets into modified targets Y^{mod} , ensuring frame-wise correspondence with the predictions.

with $\mathbf{y}_n^{\text{strong}} \in \mathbb{R}^D$ which temporally corresponds to X on the frame level, as visualized in Figure 8.1a. In the example of MPE, the target features are typically encoded as binary multi-hot vectors indicating the presence of certain pitches. In the case of strong targets, we can use an element-wise loss function to train the network, such as MSE

$$c_{\text{MSE}}(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \quad (8.1)$$

or BCE

$$c_{\text{BCE}}(\mathbf{x}, \mathbf{y}) = -\mathbf{y}^\top \log \mathbf{x} - (1 - \mathbf{y})^\top \log(1 - \mathbf{x}), \quad (8.2)$$

where the logarithm of a vector is defined element-wise.

In practice, strongly aligned targets are rarely available in MIR. Instead, weakly labeled targets, which share only a global correspondence with the input data, are more readily obtainable (Figure 8.1b). For instance, in MPE, weak targets $Y = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_M)$ with $\mathbf{y}_m \in \mathbb{R}^D$ and $m \in \{1, 2, \dots, M\}$ can be derived from note events in a musical score. While Y and Y^{strong} contain the same set of feature vectors, they differ in the number of repetitions of each vector.

To train DNNs with weak targets, an alignment step between network predictions and the weak targets is essential. To this end, SDTW has been introduced as a differentiable extension of DTW, and was applied in tasks such as MPE and PCE [83, 167]. Still, SDTW-based training exhibits instabilities, with performance variations influenced by data representation [83], training strategies (see Chapter 5 and [167]), and hyperparameter choices such as step weights (see Chapter 6 and [168]). In previous experiments in Chapter 5 we have observed issues like alignment collapse and diagonalization, but so far, there is no straightforward way to analyze the SDTW training process. One key challenge is that the network parameter updates resulting from the SDTW loss are difficult to interpret due to the algorithm's mathematical complexity. In contrast, when training with strongly aligned targets using element-wise loss functions such as MSE or BCE, the optimization process is more transparent, as the network simply minimizes the distance between predictions and strong targets.

In this chapter, we address the following fundamental questions: How does SDTW training differ from standard training with strongly aligned targets? What does a network actually learn when trained with an SDTW loss? To provide answers, as a key contribution of this chapter we reformulate the SDTW gradient into an equivalent representation derived from element-wise MSE and BCE losses. This reformulation introduces interpretable *modified targets* $Y^{\text{mod}} = (\mathbf{y}_1^{\text{mod}}, \dots, \mathbf{y}_N^{\text{mod}})$ that, when used as strong targets in an element-wise loss function, produce identical network updates to those obtained under SDTW with weak targets. In other words, the DNN learns to minimize the distance (e.g., MSE or BCE) to these modified targets. By inspecting the modified targets (Figure 8.1c), we gain insights into what the DNN actually learns, allowing us to analyze the impact of SDTW hyperparameters and training strategies on network performance. Furthermore, these modified targets can be visualized or sonified at early training stages to provide qualitative assessments of the learning process. Distance measures to strongly aligned reference targets can also be computed, facilitating quantitative evaluations.

8.2 Gradient Reformulation into Modified Targets

The SDTW gradient $\mathbf{H} = \nabla_{\mathbf{C}} \text{SDTW}(\mathbf{C})$ is usually computed via efficient DP recursions (see Chapters 4 to 7), and the gradient w.r.t. the network outputs $\mathbf{x}_n$ is obtained by applying the chain rule:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{x}_n} = \sum_{m=1}^M \mathbf{H}(n, m) \cdot \frac{\partial c(\mathbf{x}_n, \mathbf{y}_m)}{\partial \mathbf{x}_n}. \quad (8.3)$$

This gradient is typically computed using the automatic differentiation modules available in modern DL frameworks. From the formulation in (8.3), it remains unclear which features the DNN actually learns when trained with an SDTW loss. In contrast, when training with strongly aligned targets using standard element-wise loss functions such as MSE or BCE, the optimization process is more transparent. For instance, given the gradients:

$$\frac{\partial c_{\text{MSE}}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} = \mathbf{x} - \mathbf{y} \quad (8.4)$$

$$\frac{\partial c_{\text{BCE}}(\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}} = -\frac{\mathbf{y}}{\mathbf{x}} + \frac{1 - \mathbf{y}}{1 - \mathbf{x}}, \quad (8.5)$$

it is evident that the network parameters are adjusted to bring the predictions $\mathbf{x}$ closer to the targets $\mathbf{y}$. However, in the case of SDTW, the alignment process introduces additional complexity, making it less intuitive to interpret what the network is optimizing towards.

8.2.1 Derivation for MSE and BCE Cost

Next, we reformulate the SDTW gradient from (8.3) into the standard element-wise loss functions defined in (8.4) and (8.5). We then demonstrate the equivalence of the modified target representations for MSE and BCE, offering a unified perspective on SDTW-based training.

MSE as local cost

For $c = c_{\text{MSE}}$, the gradient of the SDTW cost w.r.t. the DNN predictions is given by:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{x}_n} = \sum_{m=1}^M \mathbf{H}(n, m) \cdot (\mathbf{x}_n - \mathbf{y}_m), \quad (8.6)$$

which follows from (8.3) and (8.4). Our goal is to reformulate this expression into the standard form of (8.4). We introduce the row sum of the gradient matrix:

$$h(n) := \sum_{m=1}^M \mathbf{H}(n, m) \in \mathbb{R} \quad (8.7)$$

and define the row-normalized gradient matrix as:

$$\tilde{\mathbf{H}}(n, m) := \mathbf{H}(n, m) / h(n) \in \mathbb{R}^{N \times M}. \quad (8.8)$$

Using these definitions, we can rewrite the gradient from (8.6) as:

$$\begin{aligned} \frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{x}_n} &= h(n) \cdot \mathbf{x}_n - \sum_{m=1}^M \mathbf{H}(n, m) \mathbf{y}_m \\ &= h(n) \cdot \left(\mathbf{x}_n - \sum_{m=1}^M \tilde{\mathbf{H}}(n, m) \mathbf{y}_m \right) \\ &= h(n) \cdot \left(\mathbf{x}_n - \mathbf{y}_n^{\text{mod}} \right), \end{aligned} \quad (8.9)$$

where we define the modified targets:

$$\mathbf{y}_n^{\text{mod}} := \sum_{m=1}^M \tilde{\mathbf{H}}(n, m) \cdot \mathbf{y}_m \in \mathbb{R}^D. \quad (8.10)$$

Thus, the network parameters are updated such that the predictions $\mathbf{x}_n$ move closer to the modified targets $\mathbf{y}_n^{\text{mod}}$. The update magnitude is determined by $h(n)$.

BCE as local cost

A similar reformulation applies to the SDTW gradient when using BCE as the local cost function. Starting with the gradient:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{x}_n} = \sum_{m=1}^M \mathbf{H}(n, m) \cdot \left(-\frac{\mathbf{y}_m}{\mathbf{x}_n} + \frac{1 - \mathbf{y}_m}{1 - \mathbf{x}_n} \right), \quad (8.11)$$

we rewrite this expression in the form of (8.5) as follows:

$$\frac{\partial \text{SDTW}(\mathbf{C})}{\partial \mathbf{x}_n} = -\frac{\sum_m \mathbf{H}(n, m) \mathbf{y}_m}{\mathbf{x}_n} + \frac{\sum_m \mathbf{H}(n, m) - \sum_m \mathbf{H}(n, m) \mathbf{y}_m}{1 - \mathbf{x}_n} \quad (8.12)$$

$$= h(n) \cdot \left(-\frac{\sum_m \tilde{\mathbf{H}}(n, m) \mathbf{y}_m}{\mathbf{x}_n} + \frac{1 - \sum_m \tilde{\mathbf{H}}(n, m) \mathbf{y}_m}{1 - \mathbf{x}_n} \right) \quad (8.13)$$

$$= h(n) \cdot \left(-\frac{\mathbf{y}_n^{\text{mod}}}{\mathbf{x}_n} + \frac{1 - \mathbf{y}_n^{\text{mod}}}{1 - \mathbf{x}_n} \right). \quad (8.14)$$

Notably, this reformulation uses the same modified targets $\mathbf{y}_n^{\text{mod}}$ and weighting factor $h(n)$ as in the MSE case. These modified targets can be visualized or sonified alongside the original signal, providing an intuitive way to analyze the SDTW training process. Importantly, this approach does not require knowledge of the strong reference targets, making it highly suitable for analyzing training behavior in real-world scenarios where only weakly aligned data is available.

8.2.2 Properties of the Modified Targets and Magnitude Decay

In this section, we analyze the theoretical properties of the modified targets and illustrate them using a synthetic example (Figure 8.2). Specifically, we investigate how the SDTW loss influences blurring and magnitude variations when predicting short and long features. To demonstrate these effects, we construct a sequence Y consisting of six binary feature vectors $\mathbf{y} \in \{0, 1\}^{12}$, three of which are all-zero (see Figure 8.2a). Next, we construct a sequence X (as seen in Figure 8.2b), by repeating the non-zero elements of Y two, one, and five times, respectively, and repeating the all-zero elements three times each. Note that in the case of classical DTW with hard alignments, in our example X and Y could be aligned with zero cost. We are interested in the alignment of these sequences under SDTW loss and thus compute the forward and backward passes with uniform step weights $w_h = w_v = w_d = 1$ and $\gamma = 1$ as described in Chapter 6 [168]. We then derive the alignment matrix $\tilde{\mathbf{H}}$ and compute the modified targets Y^{mod} , displaying the results in Figure 8.2c and Figure 8.2d, respectively. Ideally, as we chose X to be an unfolded version of Y without additional noise, Y^{mod} should closely resemble X .

Due to the relatively high softmax temperature $\gamma = 1$, we observe significant temporal blurring in both the modified targets and the alignment matrix. By definition, the row sums of the alignment matrix $\tilde{\mathbf{H}}$ are normalized, i.e., $\sum_{m=1}^M \tilde{\mathbf{H}}(n, m) = 1$ for all $n \in \{1, \dots, N\}$. This normalization ensures that each predicted frame is assigned a convex combination of weak target frames. However, when targets are aligned for only

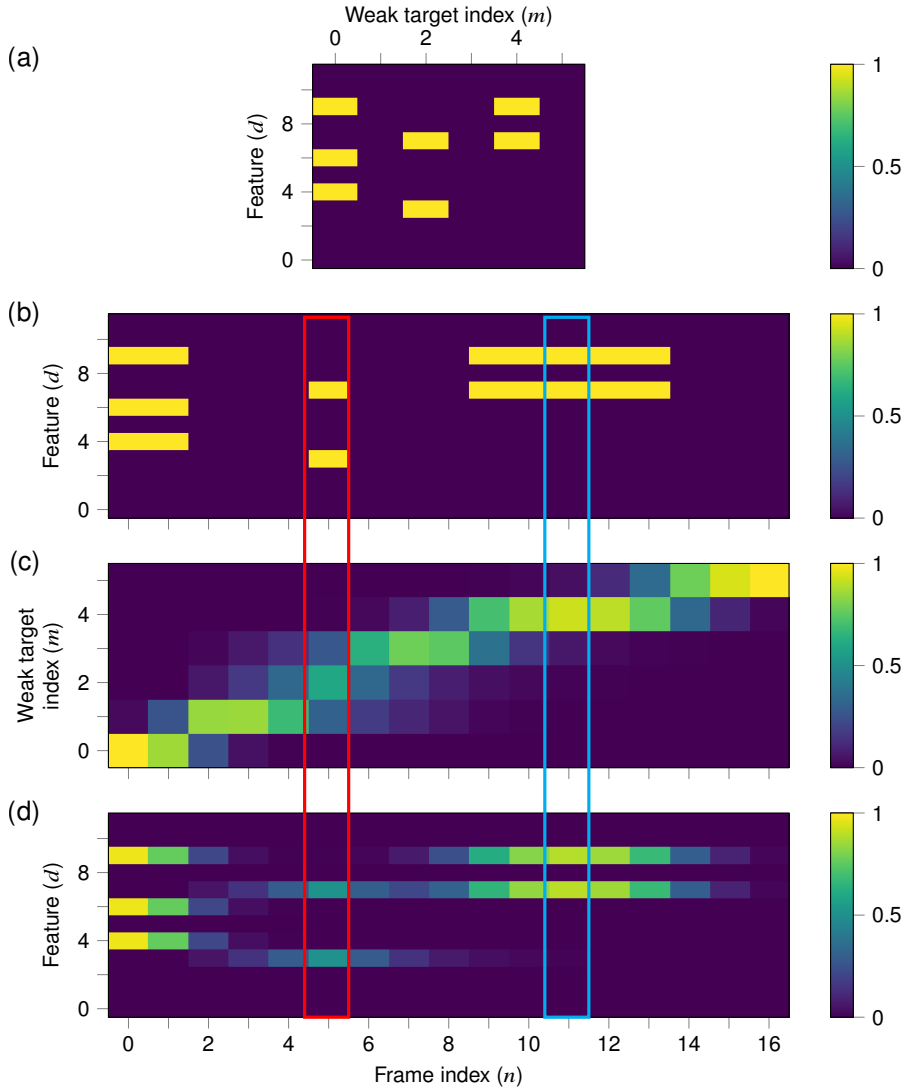


Figure 8.2: Illustration of magnitude decay in modified targets. (a) Weakly aligned target sequence Y . (b) Predicted sequence X , which corresponds to a perfectly aligned and unfolded version of Y . (c) Alignment matrix $\tilde{\mathbf{H}}^T$. (d) Modified targets Y^{mod} , showing significant variations in magnitude. A short feature instance experiencing magnitude decay is highlighted in red, while a long feature instance with near-complete magnitude preservation is highlighted in blue.

a short duration (e.g., staccato notes), the temporal blurring from neighboring frames overlaps with the actual target, leading to a significant reduction in its magnitude after normalization by $h(n)$. This effect is illustrated in Figure 8.2, where the frame marked in red shows how short events are particularly affected. Conversely, for targets aligned over a longer duration (e.g., sustained notes), the influence of temporal blurring is primarily limited to the onset and offset regions, leaving the central frames mostly unaffected. This phenomenon is visible in the section marked in blue in Figure 8.2, where the corresponding frame in Y^{mod} retains a relatively high magnitude. Consequently, with a higher softmin temperature γ , SDTW introduces a pronounced magnitude imbalance: short events tend to have reduced magnitudes, while the central frames of long events maintain higher magnitudes.

Figure 8.3: Musical score for running example.

8.3 Evaluation

In this section, we examine modified targets in a DNN training scenario. To ensure clarity and intuitive accessibility, we select MPE as a case study—a straightforward yet relevant task well-suited for providing insights into the model behavior. MPE is particularly appropriate due to its diverse weakly labeled datasets, consisting of score–audio pairs, and its broad range of target durations, from short staccato notes to sustained tones. Additionally, it provides an intuitive framework for visualization and analysis. Rather than aiming to advance the state of the art in MPE or report performance benchmarks, our objective is to demonstrate how early-stage inspection of modified targets can yield valuable insights into the learning process and serve as a reliable predictor of final model performance. Concluding this section, we integrate theoretical findings from the previous section with empirical insights from our MPE experiment to formulate practical recommendations for training state-of-the-art MIR models with SDTW. Sonifications for all presented examples and further links to Pytorch implementations are available at our website.¹²

8.3.1 Experimental Setup

As an example architecture for MPE, we adopt a single convolutional stack from the Python implementation of the *Onsets and Frames* model [63]. The stack processes a Mel spectrogram as input and consists of three convolutional layers with batch normalization, max pooling, and dropout, followed by two fully connected layers with sigmoid activation. In total, the model comprises approximately 4.3 million trainable parameters. We choose the *Onsets and Frames* model as our basis due to its widespread use in the literature and its proven effectiveness for transcription tasks. However, we simplify the architecture by using only a single stack, reducing interdependencies between multiple stacks present in the original model. This modification not only decreases the model size but also enhances interpretability by removing recurrent neural networks from the pipeline.

We pre-train the model on strongly aligned data from the MAESTRO [64] dataset for 100000 steps with audio of 20 s length and a batch size of 8, BCE loss, Adam optimizer [74] with an initial learning rate of $6 \cdot 10^{-4}$ and a reduction of the learning rate by a factor of 0.98 every 10000 steps, and gradient clipping.

We fine-tune the model using weakly aligned data from the BPSD, see Chapter 3 [170]. For this, we automatically generate training samples by pairing multi-pitch labels with corresponding audio segments,

¹² https://audiolabs-erlangen.de/resources/MIR/2025_ZeitlerM_softDTW_modTargets_ISMIR

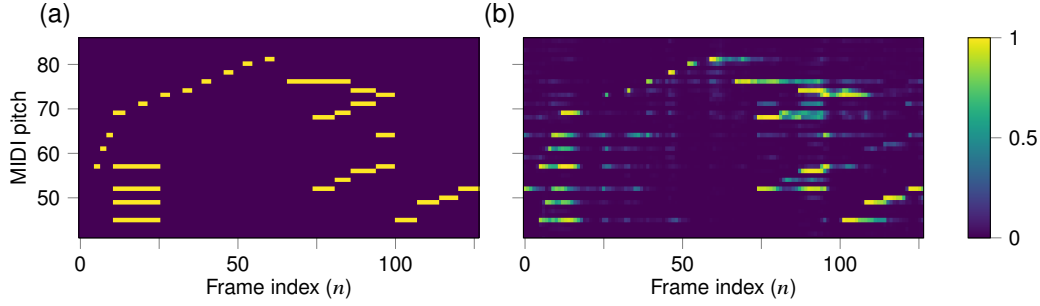


Figure 8.4: Pianoroll representation of the running example. (a) Strongly aligned reference targets. (b) Predictions of pretrained model.

each spanning 8 measures. If a segment exceeds 20 seconds in duration, it is excluded from training due to hardware memory constraints. The segments are grouped into batches of size 8. For optimization, we use the weighted SDTW loss from Chapter 6 with BCE as the local cost function. The model is trained for 5000 steps using the Adam optimizer [74] with a learning rate of 10^{-3} . We evaluate performance on a test set from the BPSD with versions that were not included in the training. The pretrained model achieves an F-measure of 0.60 on the test set.

The weak targets Y are derived by removing all repetitions from the strongly aligned target sequence Y^{strong} , which is provided in the BPSD. For additional details on the network architecture, we refer to [63], and for a comprehensive explanation of the weighted SDTW loss along with a Python implementation, we refer to Chapter 6 and [168].

8.3.2 Analyzing Modified Target Representations

We now analyze the modified targets for both the pretrained model and the final predictions after fine-tuning with the SDTW loss over 5000 training steps. Our goal is to examine how the predictions of the fine-tuned model align with the modified targets obtained from the pretrained model. For this analysis, we use an excerpt from the first movement of Beethoven’s second piano sonata (Op. 2 No. 2), as shown in Figure 8.3. The corresponding reference targets (strongly aligned) and the pretrained model’s predictions for a performance by Alfred Brendel (1996) are illustrated in Figure 8.4a and Figure 8.4b, respectively. This excerpt features a transition from a staccato passage to a legato section with sustained notes, making it a representative test case for illustrating how SDTW handles variations in note duration.

For our experiments, we use step weights of $w_h = 0.1$, $w_v = 1$, and $w_d = 1$, reducing the weight of horizontal steps (target repetition) to enhance robustness against prediction outliers (see Chapter 6 and [168]). We vary the softmin temperature $\gamma \in \{0.1, 1.0, 10.0\}$ and present the results for the modified training targets of the pretrained model and the predictions of the fine-tuned model in Figure 8.5.

For $\gamma = 0.1$ (Figure 8.5a), the modified targets align closely with the reference targets from Figure 8.4, showing only slight blurring. This leads to final predictions that capture all notes with relatively high

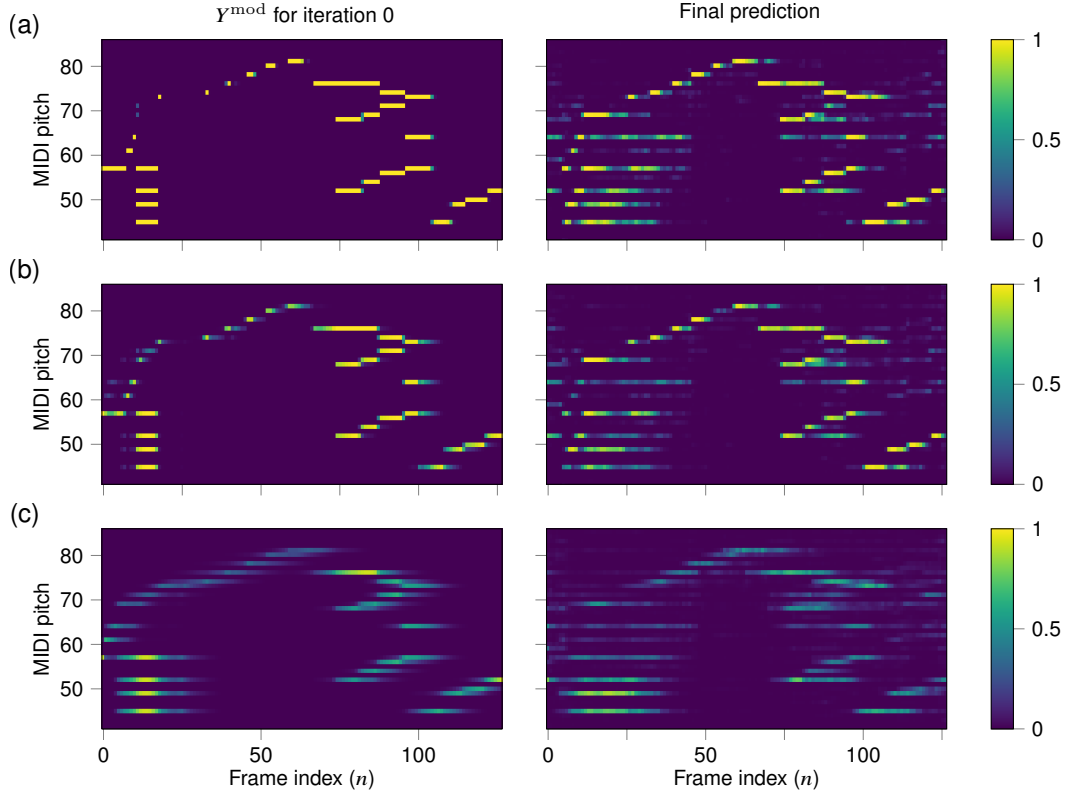


Figure 8.5: Visualization of modified targets (left) and model predictions (right) for different SDTW configurations. (a) $\gamma = 0.1$. (b) $\gamma = 1$. (c) $\gamma = 10$.

and consistent magnitude across detected events, resulting in a test F-measure of 0.67. For $\gamma = 1.0$ (Figure 8.5b), the SDTW loss causes slight blurring at the note onsets and offsets of the modified targets, accompanied by a reduced magnitude for notes in the staccato movement. As the predictions can never get better than what is given by the training targets, also the predicted notes of the fine-tuned model reveal stronger blurring and slightly lower magnitudes than for $\gamma = 0.1$. The test F-measure slightly reduces to 0.64. With $\gamma = 10.0$ (Figure 8.5c), note events in the modified targets become even more blurred. For all but the longest notes, magnitudes drop considerably, falling below 0.5 in the staccato movement. The fine-tuned model’s predictions closely follow this pattern, showing a pronounced magnitude reduction for most notes, with a test F-measure of only 0.36. Notably, the detected staccato notes fall below 0.5, which is problematic for post-processing tasks that often discard events under this threshold.

8.3.3 Practical Implications of SDTW Reformulation

In this section, we outline some practical implications of the proposed reformulation of SDTW when training DNNs from scratch. Previous studies in [82] and Chapter 6 [167] have observed that training a DNN from a poor initialization requires a relatively high softmin temperature parameter γ . On the one hand, when $\gamma \rightarrow 0$, SDTW alignments often degenerate, leading to a collapse in model training. On

the other hand, for sufficiently high γ , the network is exposed to a weighted combination of multiple alignments, which facilitates successful training initialization.

Given that a high γ is necessary to initialize DNN training with weak targets, we now examine its implications in the context of a common transcription scenario such as *Onsets and Frames* [63]. Since transcription models aim to predict discrete events (e.g., symbolic note information), a common post-processing step involves applying a detection threshold to the raw network output, treating events above the threshold as active. One direct consequence of the temporal blurring induced by high γ is that predictions fade in and out before and after the actual event. If the magnitude within these fading regions lies above the detection threshold, the thresholded predictions extend before and after the actual event, effectively widening the detected temporal span. In time-sensitive tasks such as onset estimation, this can lead to an undesirable temporal shift in post-processed predictions.

A second observed effect, both theoretically derived and empirically demonstrated, is a decay in magnitude for short events. In onset estimation, where events are inherently short, this decay can be particularly problematic. If the magnitude of short events falls below the detection threshold, these events may be lost entirely after post-processing. This issue is especially critical in models like *Onsets and Frames*, where note activation is conditioned on a preceding onset [63].

Based on these conceptual findings and in line with previous results from Chapter 5 [167], we offer the following recommendations for using SDTW in DNN training scenarios with poor initialization:

- 1. High γ for initialization:** Begin training with a relatively high softmin temperature γ to ensure stable convergence. During this phase, post-processed performance metrics (e.g., note accuracy in transcription) may be unreliable due to temporal shifting and magnitude decay.
- 2. Target inspection:** Despite the shortcomings of metrics after post-processing, the modified targets can be inspected at any point to verify whether the DNN is learning meaningful patterns.
- 3. Gradual γ reduction:** After the initialization phase, progressively lower γ until the modified targets exhibit less temporal blurring and a balanced magnitude for both short and long events.
- 4. Resume training:** With refined SDTW parameters, continue training to allow predictions to converge toward the improved modified targets.

8.4 Further Notes

In this chapter, we introduced a reformulation of the SDTW gradient into interpretable modified targets, which yield identical network parameter updates when used with standard element-wise loss functions. Through theoretical analysis and a controlled experiment, we demonstrated that temporal blurring and magnitude decay are inherently part of training with SDTW, even though it is not visible in the underlying weak targets. By making the training process more transparent, our approach provides researchers and practitioners with deeper insights into SDTW-based learning and offers an intuitive, practical method for analyzing weakly supervised training strategies.

9 Unifying Views: CTC and Differentiable DTW

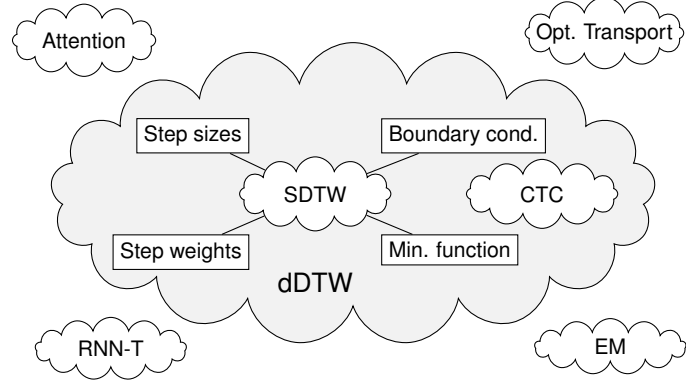
This chapter is based on [165]. Johannes Zeitler is the main contributor to this article. He developed the research concept, implemented and conducted the experiments, and prepared the manuscript. Meinard Müller supervised the work and contributed throughout all stages of research and manuscript preparation.

After developing several extensions of differentiable alignment, we now step back to analyze their relationship to existing sequence learning objectives. Apart from SDTW-based loss functions, Connectionist Temporal Classification (CTC) is a widely used training objective for optimizing sequence models on weakly aligned data, particularly in settings where targets consist of discrete labels. While CTC is effective and stable, it is structurally rigid and relies on a special blank symbol that is introduced to model target repetitions but behaves in a way that is hard to interpret.

In this chapter we establish a formal connection between CTC and SDTW and introduce Differentiable Dynamic Time Warping (dDTW), a generalized differentiable alignment framework that incorporates flexible step sizes, step weights, boundary conditions, and minimum functions. Within this unified framework, CTC, SDTW, and variants thereof emerge as special cases. Through controlled experiments on representative MIR tasks, including single- and multi-label classification settings, we compare prediction accuracy, training stability, alignment behavior, and the role of the blank symbol. With these results we demonstrate that CTC and SDTW are not fundamentally distinct alignment paradigms, but they can be viewed as special cases of a unifying dDTW framework.

The remainder of this chapter is structured as follows. We start in Section 9.1 with an overview of differentiable alignment algorithms, putting CTC and SDTW in a larger context. In Sections 9.2 and 9.3, we briefly review the objectives of CTC and SDTW, respectively, focusing on their global formulations rather than algorithmic specifics. In Section 9.4, we present our first main contribution: a formal transformation between the CTC and SDTW objectives, along with the modifications to SDTW required to achieve equivalence. As our second main contribution, we introduce in Section 9.5 the dDTW algorithm, which incorporates these extensions to SDTW. We detail its building blocks and provide efficient DP recursions for the loss and gradient computation. We specify label probabilities and cost functions in Section 9.6 and experimentally evaluate in Section 9.7 performance of the CTC, SDTW, and combinations

Figure 9.1: Schematic illustration of alignment-based loss functions. The dDTW framework generalizes the CTC and SDTW algorithms.



thereof in the unified dDTW framework. Finally, we summarize our findings and outline future research directions in Section 9.8.

9.1 Motivation and Related Work

A widely adopted method for end-to-end DNN training with weak supervision is the Connectionist Temporal Classification (CTC) loss [59], which provides a differentiable mechanism to align prediction and label sequences during loss computation. Originally introduced for sequence labeling tasks, CTC has since become a standard loss function in areas such as optical character and music recognition [132] and ASR [4]. In MIR, CTC has been successfully applied to tasks such as score–audio retrieval [160] and lyrics alignment [135]. To address multi-label scenarios, the classical CTC algorithm has been extended to Multi-Label Connectionist Temporal Classification (MCTC) [156], and successfully applied to multi-pitch and pitch class estimation tasks [146, 147]. While CTC is known for its algorithmic stability and the potential to train large models from scratch using unaligned data, it also presents several limitations. Specifically, the formulation can be unintuitive, especially in the multi-label setting, structurally inflexible, and reliant on a special blank symbol. This blank symbol is designed to model target repetitions and implicitly stabilize the alignment process, but introduces architectural constraints and often leads to spiky, blank-dominated predictions during inference [59, 160, 171]. Furthermore, CTC is inherently limited to feature-to-label tasks with targets from a finite alphabet.

Similarly, Soft Dynamic Time Warping (SDTW) [30, 94] offers a differentiable generalization of classical DTW [98]. By replacing the hard minimum operator with a smooth approximation, SDTW remains conceptually simple and interpretable. Several extensions have been proposed for SDTW, including alternative smoothing strategies [61] and improvements regarding mathematical properties [17]. This thesis advanced SDTW by introducing training stabilization techniques (see Chapter 5), customizable step weights (see Chapter 6), and relaxed boundary conditions (see Chapter 7). Unlike CTC, SDTW and its extensions operate on arbitrary cost matrices and are applicable to diverse problem settings, including

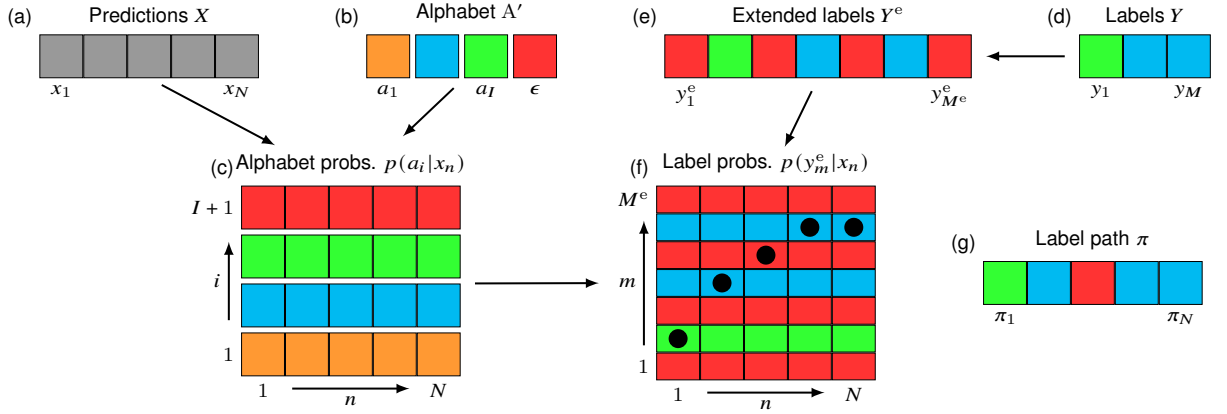


Figure 9.2: Overview of CTC loss computation. (a) Prediction sequence X . (b) Extended alphabet A' . (c) Probabilities of alphabet symbols $p(a_i|x_n)$. (d) Label sequence Y . (e) Extended label sequence Y^e . (f) Probabilities of extended labels $p(y_m^e|x_n)$. (g) Possible label path π corresponding to circles in (f). Colors represent symbols and color gradients represent time steps.

feature-to-label tasks via one-hot and multi-hot encodings, as well as feature-to-feature alignment tasks using continuous-valued feature representations [40, 82].

In this chapter, we establish a formal connection between CTC and SDTW by reformulating the CTC objective to match a generalized SDTW formulation. To achieve mathematical equivalence between the two losses, we extend SDTW with three building blocks: parameterizable cell-wise step weights, adjustable step sizes, and flexible boundary conditions (see Figure 9.1). We refer to this generalized alignment algorithm as Differentiable Dynamic Time Warping (ddTW) and derive efficient DP recursions for both the forward and backward passes. The proposed framework explicitly relates CTC and SDTW, with both being special cases of ddtw. It offers a more interpretable alternative to CTC while removing several of its limitations, most notably, the need for a special blank symbol. In an illustrative and controlled experiment—specifically, pitch class estimation from music recordings—we compare CTC and SDTW within the common ddtw framework, evaluating prediction accuracy, training stability, alignment behavior, and discussing the impact of and alternatives to the blank symbol. Our results show that a well-parameterized ddtw loss retains the stability of CTC, eliminates the need for a blank symbol, and achieves even higher prediction accuracy. These findings offer a basis for differentiable alignment techniques beyond the classical CTC loss.

9.2 CTC Revisited

In this section, we describe the CTC objective function as proposed by Graves et al. [59]. Our aim is to gain a conceptual understanding of CTC, rather than focusing on algorithmic details. Therefore, we limit the description to the global loss objective of CTC and refer the reader to the original publication [59] for implementation details regarding, e.g., the forward and backward passes using DP.

9.2.1 Alphabet, Labels, and Predictions

Following the original CTC paper [59], we define an **alphabet** $A = \{a_1, \dots, a_I\}$ of available symbols, and extended alphabet $A' = A \cup \{\epsilon\}$ including a special blank symbol ϵ . Let $Y = (y_1, \dots, y_M)$ denote a **label sequence** with labels $y_m \in A$. The label sequence represents the symbols in their correct order but does not encode any information about the duration of individual symbols. For example, in ASR, a label sequence may correspond to a phoneme sequence, while in music transcription it may represent a sequence of note events derived from a musical score. As the CTC requires a special blank symbol, we follow [59] by adding the blank symbol before and after every element in Y , resulting in the **extended label sequence** $Y^e = (\epsilon, y_1, \epsilon, \dots, \epsilon, y_M, \epsilon) = (y_1^e, \dots, y_{M^e}^e)$ with $M^e = 2M + 1$ and $y_m^e \in A'$. Furthermore, we define output features, also called **prediction sequence** $X = (x_1, \dots, x_N)$ with $x_n \in \mathcal{F}_X$, where $\mathcal{F}_X$ denotes an abstract feature space, e.g., symbol probabilities.

9.2.2 Label Paths

Our goal is to align the label sequence to the prediction sequence. Following [59], we establish notation for alignments in terms of **label paths** $\pi = (\pi_1, \dots, \pi_N)$, which model the alignment as a sequence of label symbols from the extended alphabet with $\pi_n \in A'$. To define a validity condition for a label path, a projection operator $\kappa_N : A'^N \rightarrow A^{\leq N}$ is defined that takes a label path π and first collapses repeated symbols and then removes all blank symbols. A **valid** label path must, by definition, satisfy $\kappa_N(\pi) = Y$ and can be constructed from the extended label sequence Y^e , with the following rules:

- **Start condition:** The first path element π_1 is either the blank symbol ϵ or the first label y_1 ; in other words, $\pi_1 \in \{y_1^e, y_2^e\}$.
- **End condition:** The last path element π_N is either the blank symbol ϵ or the last label y_M ; in other words, $\pi_N \in \{y_{M^e-1}^e, y_{M^e}^e\}$.
- **Step sizes:** The path π is a (weakly) monotonic unfolding of Y^e . Leaving out blank symbols is allowed if the preceding and succeeding labels are different. In other words, let $\pi_n = y_m^e$, then

$$\begin{aligned} \pi_{n+1} &\in \{y_m^e, y_{m+1}^e, y_{m+2}^e\}, & \text{if } y_m^e \neq y_{m+2}^e, \\ \pi_{n+1} &\in \{y_m^e, y_{m+1}^e\}, & \text{else.} \end{aligned} \tag{9.1}$$

The set of all valid label paths of length N that collapse to a label sequence Y is denoted by $\kappa_N^{-1}(Y)$.

9.2.3 CTC Objective

Following [59] and assuming conditional independence of predictions and labels over time, we define the probability of a label path π given a prediction sequence X as

$$p(\pi|X) = \prod_{n=1}^N p(\pi_n|x_n), \quad (9.2)$$

where $p(\pi_n|x_n)$ denotes the probability of label π_n given the feature vector x_n . We will give a concrete example of this probability in Section 9.6. The probability of the label sequence Y is obtained by summing the probabilities of all valid label paths $\pi \in \kappa_N^{-1}(Y)$ that project to Y :

$$p(Y|X) = \sum_{\pi \in \kappa_N^{-1}(Y)} p(\pi|X). \quad (9.3)$$

While the direct evaluation of (9.3) has computational complexity $\mathcal{O}(|\kappa_N^{-1}(Y)| \cdot N)$, in practice the CTC loss is computed efficiently using a dynamic programming (DP) recursion with complexity $\mathcal{O}(M^e \cdot N)$ [59].

9.2.4 Practical Implementation

In standard CTC implementations, the process starts by computing probabilities for all symbols in the extended alphabet A' conditioned on the network predictions X (Figure 9.2a-c). The label sequence (Figure 9.2d) is then expanded by inserting blanks before and after each label (Figure 9.2e), and rows from the alphabet probability matrix are arranged accordingly (Figure 9.2f). The loss is computed on this re-arranged matrix by accumulating the probabilities of all valid label paths π (Figure 9.2g). For large alphabets (e.g., in multi-pitch estimation), computing probabilities for all symbols becomes computationally expensive (Figure 9.2c). This can be mitigated by restricting the alphabet to only active labels at each step, reducing complexity, however, at the cost of additional preprocessing.

9.3 SDTW Revisited

In this section, we briefly describe the SDTW loss function as proposed by Cuturi et al. [30] by defining the cost matrix as an input to the algorithm, specifying alignment paths, and outlining the global objective of SDTW.

9.3.1 Labels, Predictions, and Cost Matrix

The computation of the SDTW loss is based on a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ (Figure 9.3c) that is computed from two sequences $X = (x_1, \dots, x_N)$ (Figure 9.3a) and $Y = (y_1, \dots, y_M)$ (Figure 9.3b) [30] with $x_n \in \mathcal{F}_X$ and $y_m \in \mathcal{F}_Y$ (e.g., $\mathcal{F}_Y = \mathcal{A}'$). The elements of the cost matrix are defined as

$$\mathbf{C}(n, m) = c(\mathbf{x}_n, \mathbf{y}_m), \quad (9.4)$$

where $c : \mathcal{F}_X \times \mathcal{F}_Y \rightarrow \mathbb{R}$ is a local cost function, e.g., BCE. In the next section, we define alignment paths over the cost matrix.

9.3.2 Alignment Paths

While the label path introduced for CTC is defined w.r.t. elements in the alphabet $\mathcal{A}'$, we can alternatively define an equivalent **alignment path** $P = (p_1, \dots, p_L)$ that indexes a sequence of cost matrix cells $p_\ell = (n_\ell, m_\ell) \in \mathcal{I}$ for $\ell \in [1 : L]$, where

$$\mathcal{I} := [1 : N] \times [1 : M] \quad (9.5)$$

denotes the set of all cells in the cost matrix. Analogously, the index pairs (n_ℓ, m_ℓ) provide a mapping between the two feature sequences X and Y , as known from classical DTW [98]. Valid alignment paths follow a set of constraints, in particular boundary and step size conditions. We specify boundary conditions $\mathcal{B}_{\text{start}}$ and $\mathcal{B}_{\text{end}}$ that define the allowed start and end points for the alignment path. The boundary conditions are given by a set of integer pairs, i.e., $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$. For any alignment path P , it must hold that $p_1 \in \mathcal{B}_{\text{start}}$ and $p_L \in \mathcal{B}_{\text{end}}$ such that the boundary conditions are fulfilled. We define the allowed step sizes $\mathcal{S} = \{(i_s, j_s) \mid s \in [1 : S]\}$ as a set of integer tuples with monotonicity constraint $i_s \geq 0, j_s \geq 0, i_s + j_s \geq 1$ for all $(i_s, j_s) \in \mathcal{S}$. For all cells in the alignment path, it must hold that $p_\ell - p_{\ell-1} \in \mathcal{S}$. For SDTW, a valid alignment path satisfies the following constraints [30]:

- **Start condition:** The first element of the alignment path aligns the first elements of the prediction and target sequences, i.e., $\mathcal{B}_{\text{start}} = \{(1, 1)\}$.
- **End condition:** The last element of the alignment path aligns the last elements of the prediction and target sequences, i.e., $\mathcal{B}_{\text{end}} = \{(N, M)\}$.
- **Step sizes:** The possible steps are restricted to $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$.

Let $\mathcal{P}$ denote the set of all valid alignment paths for two sequences of length N and M , satisfying boundary and step size conditions. The cardinality of $\mathcal{P}$ is given by the Delannoy number [98].

9.3.3 SDTW Objective

The SDTW objective is to find the alignment path of minimum total cost over the matrix $\mathbf{C}$, restricted to valid paths $P \in \mathcal{P}$. In its global formulation, the SDTW cost is defined as the soft minimum of the total costs over all valid alignment paths [30, 94]:

$$\begin{aligned} \text{SDTW}(\mathbf{C}) &= \mu_{\text{soft}}^{\gamma} \left(\left(\sum_{(n,m) \in P} \mathbf{C}(n,m) \right)_{P \in \mathcal{P}} \right) \\ &= -\gamma \log \left(\sum_{P \in \mathcal{P}} \exp \left(\sum_{(n,m) \in P} -\mathbf{C}(n,m) / \gamma \right) \right), \end{aligned} \quad (9.6)$$

where $\mu_{\text{soft}}^{\gamma}$ is the softmin function [21, 30].

9.4 Relating CTC and SDTW

In this section, we establish a formal connection between the objectives of CTC and SDTW. First, in Section 9.4.1, we reformulate the CTC and SDTW losses to an identical expression. Next, in Section 9.4.2, we define necessary requirements on the SDTW alignment paths to achieve mathematical equivalence.

9.4.1 Reformulation of CTC and SDTW

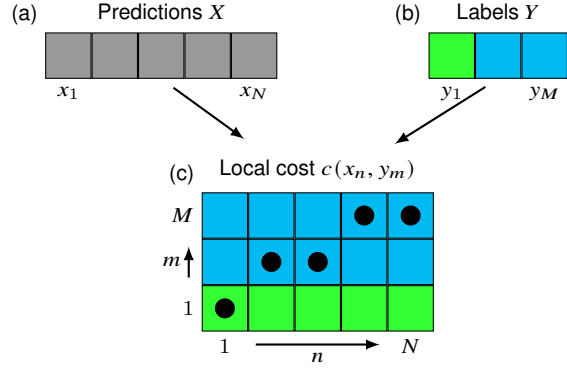
To view CTC and SDTW from a unified perspective, we first define the CTC loss $\mathcal{L}_{\text{CTC}}(X, Y)$ as the negative log-likelihood of the CTC objective, which is the standard formulation in DNN training practice [59]. Starting from (9.2) and (9.3), we obtain $\mathcal{L}_{\text{CTC}}$, which we then bring to a log-sum-exp formulation by introducing redundant exponential and log terms:

$$\begin{aligned} \mathcal{L}_{\text{CTC}}(X, Y) &= -\log p(Y|X) \\ &= -\log \left(\sum_{\pi \in \kappa_N^{-1}(Y)} \prod_{n=1}^N p(\pi_n | x_n) \right) \\ &= -\log \left(\sum_{\pi \in \kappa_N^{-1}(Y)} \prod_{n=1}^N \exp(\log p(\pi_n | x_n)) \right) \\ &= -\log \left(\sum_{\pi \in \kappa_N^{-1}(Y)} \exp \left(\sum_{n=1}^N \log p(\pi_n | x_n) \right) \right). \end{aligned} \quad (9.7)$$

In a next step, we first define the SDTW cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M^e}$ over the prediction sequence X and the extended label sequence Y^e as

$$\mathbf{C}(n, m) = -\log p(y_m^e | x_n). \quad (9.8)$$

Figure 9.3: Overview of SDTW loss computation. (a) Prediction sequence X . (b) Label sequence Y . (c) Cost matrix $\mathbf{C}$. A possible alignment path is illustrated with circles. Colors represent labels and color gradients represent time steps.



Last, we replace label paths $\pi \in \kappa_N^{-1}(Y)$ with suitably designed alignment paths $P \in \mathcal{P}$ that index elements in the prediction sequence X and extended label sequence Y^e :

$$\begin{aligned}
 \mathcal{L}_{\text{CTC}}(X, Y) &= -\log \sum_{P \in \mathcal{P}} \exp \left(\sum_{(n,m) \in P} \log p(y_m^e | x_n) \right) \\
 &= -\gamma \log \sum_{P \in \mathcal{P}} \exp \left(\sum_{(n,m) \in P} -\mathbf{C}(n, m) / \gamma \right) \\
 &= \text{SDTW}(\mathbf{C}) ,
 \end{aligned} \tag{9.9}$$

where equality holds for $\gamma = 1$. Note that this reformulation introduces new requirements for the set of alignment paths $\mathcal{P}$, as they need to exactly replicate the properties of label paths π used in CTC. We give an overview of these requirements in the following section.

9.4.2 Necessary Properties for Equivalence of CTC and SDTW

To ensure mathematical equivalence in (9.9), all alignment paths P must fulfill certain properties, as illustrated in Figure 9.4. In particular, the CTC algorithm aligns to the first predicted element either the first element of the label sequence or the first blank symbol. This modifies the start boundary condition to $\mathcal{B}_{\text{start}} = \{(1, 1), (1, 2)\}$. Likewise, the last element of the prediction sequence is aligned to the last element in the label sequence or the last blank symbol, resulting in $\mathcal{B}_{\text{end}} = \{(N, M^e - 1), (N, M^e)\}$. Considering the step sizes, the CTC algorithm is strictly monotonic over the prediction sequence (step sizes $(1, 0)$ and $(1, 1)$), and permits the skipping of blank symbols (step size $(1, 2)$) when the adjacent labels are different. This results in a set of allowed alignment step sizes $\mathcal{S} = \{(1, 0), (1, 1), (1, 2)\}$, where the $(1, 2)$ step is only permitted when skipping the blank label in the case of non-repeating adjacent labels.

9.4.3 Conclusion

While the SDTW formulation from (9.6) can be evaluated for arbitrary alignment paths P including the modified boundary and step size conditions presented in Section 9.4.2, the exhaustive search over all possible alignment paths $\mathcal{P}$ is computationally inefficient. To address this, efficient DP recursions have been proposed [30], which yield the optimal result of the global formulation (9.6) when using the softmin approximation as a differentiable minimum function. However, the original SDTW formulation from Cuturi and Blondel [30] enforces fixed start and end boundary conditions at $(1, 1)$ and (N, M) , respectively, and restricts step sizes to the set $\{(1, 0), (0, 1), (1, 1)\}$. In contrast, Mensch and Blondel [94] propose a differentiable DP framework for general weighted directed acyclic graphs, encompassing algorithms such as Viterbi [145], SDTW [30], and attention mechanisms [143]. However, their work does not explicitly detail an efficient DP algorithm tailored for an SDTW variant with the extended step sizes and boundary conditions required for exact equivalence to CTC.

9.5 Differentiable Dynamic Time Warping

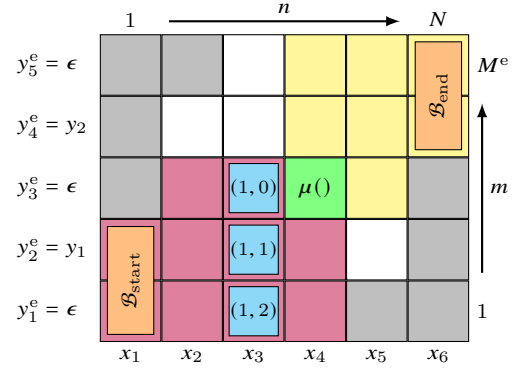
In the following, we introduce Differentiable Dynamic Time Warping (ddTW), an extension of SDTW [30] that supports flexible boundary conditions (as introduced in Chapter 7), arbitrary step sizes [94] and step weights (extending Chapter 6 and [168]), and general minimum functions [94]. The formulation adopts standard DTW conventions and employs a mathematically convenient vector–matrix notation. In this section, we assume label sequences of length M (or M^e in the case of CTC) and prediction sequences of length N , resulting in a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$. We specify a partial order $\prec$ on the set $\mathcal{I}$ of cost matrix cells, indicating precedence between cells:

$$(n, m) \prec (n', m') \quad \text{if} \quad n \leq n', m \leq m', (n, m) \neq (n', m'). \quad (9.10)$$

9.5.1 dDTW Building Blocks

Our proposed dDTW algorithm is based on efficient forward and backward DP recursions, building upon [30]. We define the dDTW algorithm with four building blocks, namely the boundary conditions, the allowed step sizes, the step weights associated with the step sizes, and the minimum functions. By choosing these building blocks accordingly, a variety of alignment algorithms can be generalized. In the following section, we define these building blocks to generalize CTC and SDTW. An illustration of the dDTW building blocks replicating CTC is provided in Figure 9.4.

Figure 9.4: Schematic overview of accumulated cost updates in the dDTW forward pass, parameterized to replicate CTC. For updating the green cell, μ is computed over prior steps (blue). Boundary conditions $\mathcal{B}_{\text{start}}$, $\mathcal{B}_{\text{end}}$ are shown in orange. All purple cells must be updated before evaluating μ ; yellow cells depend on the green cell; gray cells are unreachable under the CTC parameterization.



Boundary Conditions

We specify boundary conditions as a set of cells that define the allowed start and end points for the alignment path. In the SDTW case, the start boundary condition contains only the first sequence elements, i.e., $\mathcal{B}_{\text{start}} = \{(1, 1)\}$, whereas in the CTC parameterization it contains the first blank symbol and the first label, i.e., $\mathcal{B}_{\text{start}} = \{(1, 1), (1, 2)\}$. The end boundary condition contains in the SDTW case only last sequence elements, i.e., $\mathcal{B}_{\text{end}} = \{(N, M)\}$, and in the CTC case the last label of the extended label sequence and the blank symbol, i.e., $\mathcal{B}_{\text{end}} = \{(N, M - 1), (N, M)\}$.

Step Sizes

We define the allowed step sizes as a set of integer tuples. In the case of standard SDTW, we have $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$, and in the case of CTC we have $\mathcal{S} = \{(1, 0), (1, 1), (1, 2)\}$. In the CTC scenario it is necessary to only allow the step $(1, 2)$ in certain cells. While it is possible to define individual step sizes for each cell, we choose a different approach. We define a common set of step sizes $\mathcal{S}$ for all cells and use cell-dependent step weights, where setting a weight to infinity “blocks” forbidden steps from certain cells.

Step Weights

While individual weights associated with each step size were proposed for weighted SDTW (see Chapter 6 and [168]), we extend this idea to cell-dependent step weights for all steps. We denote the step weights for all cells by a tensor $\mathbf{W} \in \mathbb{R}_+^{N \times M \times S}$, where the element $\mathbf{W}(n, m) \in \mathbb{R}_+^S$ is a vector that contains for a cell (n, m) the weights associated to the step sizes $\mathcal{S}$. For instance, in the case of CTC, the weight tensor assigns a value of one to all permitted steps—specifically, $(1, 0)$ and $(1, 1)$ are always allowed, while $(1, 2)$ is permitted only if the skipped symbol is a blank and the adjacent symbols differ—and infinity to disallowed steps. In the parameterization for SDTW, the weight tensor is an all-ones tensor.

Algorithm 9.1 DP forward pass for dDTW

```

1: Inputs:
2: Cost matrix  $\mathbf{C} \in \mathbb{R}^{N \times M}$  indexed by cells  $\mathcal{I}$ 
3: Boundary conditions  $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$ 
4: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
5: Step weights  $\mathbf{W} \in \mathbb{R}_+^{N \times M \times S}$ 
6: Minimum function  $\mu : \mathbb{R}^S \rightarrow \mathbb{R}$ 
7: for  $m = 1, \dots, M$  do
8:   for  $n = 1, \dots, N$  do
9:     for  $s = 1, \dots, S$  do
10:       $\mathbf{f}_s^{(n,m)} = \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I}, \\ \mathbf{D}(n - i_s, m - j_s) + \mathbf{W}(n, m, s)\mathbf{C}(n, m), & \text{else,} \end{cases}$ 
11:    end for
12:     $\mathbf{f}_{S+1}^{(n,m)} = \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}}, \\ \mathbf{C}(n, m), & \text{else.} \end{cases}$ 
13:     $\mathbf{D}(n, m) = \mu(\mathbf{f}^{(n,m)})$ 
14:    Save  $\mathbf{B}(n, m) = \nabla \mu(\mathbf{f}^{(n,m)})$  for backward
15:  end for
16: end for
17:  $\mathbf{b} = (\mathbf{D}(n, m))_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
18:  $\text{dDTW}(\mathbf{C}) = \mu(\mathbf{b})$ 
19: Output: dDTW cost  $\text{dDTW}(\mathbf{C}) \in \mathbb{R}$ 

```

Minimum Function

We denote by μ a differentiable approximation of the minimum function. Note that dDTW can be used with arbitrary differentiable minimum functions, such as the “sparsemin” function [16, 94]. However, in this chapter, we concentrate on the soft minimum function defined in Section 4.2 and set $\mu = \mu_{\text{soft}}^\gamma$. Having defined the building blocks of dDTW, the following sections define how these building blocks can be integrated within the DP forward and backward recursions.

9.5.2 Forward Recursion

The dDTW algorithm takes as input a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$, calculated as described in (9.4). Following [30], we approach the task of finding the minimum cost path through the cost matrix $\mathbf{C}$ by iteratively computing a matrix of accumulated costs $\mathbf{D} \in \mathbb{R}^{N \times M}$ with a DP algorithm.

To this end, for every cell $(n, m) \in \mathcal{I}$, we define a vector of accumulated incoming cost $\mathbf{f}^{(n,m)} \in \mathbb{R}^{S+1}$, with elements $[1 : S]$ used for the cost of incoming steps $(i_s, j_s) \in \mathcal{S}$:

$$\mathbf{f}_s^{(n,m)} := \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I}, \\ \mathbf{D}(n - i_s, m - j_s) + \mathbf{W}(n, m, s)\mathbf{C}(n, m), & \text{else,} \end{cases} \quad (9.12)$$

and element $S + 1$ used for cost from a start boundary condition:

$$\mathbf{f}_{S+1}^{(n,m)} := \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}}, \\ \mathbf{C}(n, m), & \text{else.} \end{cases} \quad (9.13)$$

Evaluating the minimum function over cost values from incoming steps and potential start cells, we obtain the dDTW forward recursion as

$$\mathbf{D}(n, m) = \mu \left(\mathbf{f}^{(n,m)} \right). \quad (9.14)$$

Note that the iteration requires the computation for cells in increasing order, i.e., cell (n, m) can only be evaluated if all cells (n', m') with $(n', m') \prec (n, m)$ have been previously computed. For later use in the backward recursion, we save the gradient of the minimum function w.r.t. its input in a backtracking tensor $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ with entries

$$\mathbf{B}(n, m) := \nabla \mu \left(\mathbf{f}^{(n,m)} \right) \in \mathbb{R}^{S+1}, \quad (9.15)$$

where the indices $[1 : S]$ correspond to incoming steps, and $S + 1$ corresponds to start boundary conditions. After updating all cells $(n, m) \in \mathcal{I}$, we collect all accumulated cost values at cells given by the end boundary condition in a set $\mathbf{b} := (b^{(n,m)} \in \mathbb{R})_{(n,m) \in \mathcal{B}_{\text{end}}}$ with elements $b^{(n,m)} = \mathbf{D}(n, m)$. The total cost dDTW $(\mathbf{C}) \in \mathbb{R}$ is given by evaluating the minimum function μ over the cost at the end boundary cells specified by $\mathbf{b}$:

$$\text{dDTW}(\mathbf{C}) := \mu(\mathbf{b}). \quad (9.16)$$

The complete forward algorithm for dDTW is summarized in Algorithm 9.1.

9.5.3 Backward Recursion

Our goal is to obtain the gradient of the total cost dDTW $(\mathbf{C})$ with respect to the local cost matrix $\mathbf{C}$, defined as $\mathbf{H} := \nabla_{\mathbf{C}} \text{dDTW}(\mathbf{C}) \in \mathbb{R}^{N \times M}$. We interpret the partial derivative $\mathbf{H}(n, m) := \partial \text{dDTW}(\mathbf{C}) / \partial \mathbf{C}(n, m)$ as the “contribution” of cell (n, m) to the total alignment cost. To calculate these contributions iteratively, we “reverse” the forward recursion [30]. This concept is conceptually similar to the backtracking of classical DTW [98].

First, we apply the chain rule to separate the contribution of cost $\mathbf{C}(n, m)$ to the accumulated cost $\mathbf{D}(n, m)$ (denoted by $\mathbf{G}(n, m)$), and the contribution of accumulated cost $\mathbf{D}(n, m)$ to the total cost (denoted by

Algorithm 9.2 DP backward pass for dDTW

```

1: Inputs:
2: Backtracking tensor  $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ 
3: End boundary conditions  $\mathcal{B}_{\text{end}} \subseteq \mathcal{I}$ 
4: End boundary costs  $\mathbf{b} := (b^{(n,m)})_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
5: Step weights  $\mathbf{W} \in \mathbb{R}_+^{N \times M \times S}$ 
6: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
7: for  $m = M, \dots, 1$  do
8:   for  $n = N, \dots, 1$  do
9:     for  $s = 1 \dots S$  do
10:       $\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{E}(n+i_s, m+j_s), & \text{else.} \end{cases}$ 
11:       $\frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n, m)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{B}(n+i_s, m+j_s, s), & \text{else.} \end{cases}$ 
12:     end for
13:      $[\nabla \mu(\mathbf{b})]^{(n,m)} = \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n,m)}, & \text{if } (n, m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases}$ 
14:      $\mathbf{E}(n, m) = \sum_{s=1}^S \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} \cdot \frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n, m)} + [\nabla \mu(\mathbf{b})]^{(n,m)}$ 
15:     for  $s=1, \dots, S+1$  do
16:       $\frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n, m)} = \begin{cases} \mathbf{W}(n, m, s), & \text{if } s \in [1 : S], \\ 1, & \text{if } s=S+1 \text{ and } (n, m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases}$ 
17:     end for
18:      $\mathbf{G}(n, m) = \sum_{s=1}^{S+1} \mathbf{B}(n, m, s) \cdot \frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n, m)}$ 
19:      $\mathbf{H}(n, m) = \mathbf{E}(n, m) \cdot \mathbf{G}(n, m)$ 
20:   end for
21: end for
22: Output: Gradient  $\mathbf{H} \in \mathbb{R}^{N \times M}$ 

```

$\mathbf{E}(n, m)$) [30]:

$$\begin{aligned}
\mathbf{H}(n, m) &:= \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{C}(n, m)} \\
&= \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n, m)} \cdot \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)} \\
&= \mathbf{E}(n, m) \cdot \mathbf{G}(n, m),
\end{aligned} \tag{9.17}$$

where we define $\mathbf{E}(n, m) := \partial \text{dDTW}(\mathbf{D}) / \partial \mathbf{D}(n, m)$ and $\mathbf{G}(n, m) := \partial \mathbf{D}(n, m) / \partial \mathbf{C}(n, m)$. To compute $\mathbf{E}, \mathbf{G} \in \mathbb{R}^{N \times M}$, we again iterate over all cells $(n, m) \in \mathcal{I}$. In the backward recursion, the iteration is executed in reverse order as compared to the forward recursion, i.e., cell (n, m) can only be evaluated if all cells (n', m') with $(n, m) \prec (n', m')$ have been previously computed. For calculating the matrix $\mathbf{E}$, we use

the chain rule to reformulate:

$$\mathbf{E}(n, m) = \sum_{s=1}^S \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} \cdot \frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} + [\nabla \mu(\mathbf{b})]^{(n, m)}. \quad (9.18)$$

The first term in (9.18) denotes influence of a following cell on the total cost and, as in standard SDTW [30], is given by previously computed entries of the $\mathbf{E}$ -matrix:

$$\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{E}(n + i_s, m + j_s), & \text{else.} \end{cases} \quad (9.19)$$

The second term in (9.18), also found in standard SDTW [30], denotes influence of the current cell on a following cell and is given by the previously computed backtracking matrix:

$$\frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{B}(n + i_s, m + j_s, s), & \text{else.} \end{cases} \quad (9.20)$$

The third term in (9.18) denotes potential contributions coming from the end boundary conditions:

$$[\nabla \mu(\mathbf{b})]^{(n, m)} = \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n, m)}, & \text{if } (n, m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases} \quad (9.21)$$

In other words, (9.18) describes the influence of a cell's accumulated cost $\mathbf{D}(n, m)$ on the total alignment cost $\text{dDTW}(\mathbf{C})$ by summing up the influence of a cell (n, m) on its children $(n + i_s, m + j_s)$ weighted with the influence of the children $(n + i_s, m + j_s)$ on the final cost $\text{dDTW}(\mathbf{C})$, and an additional term $[\nabla \mu(\mathbf{b})]^{(n, m)}$ describing the influence of a path ending in (n, m) on the overall dDTW cost. This expression can be thought of as a soft version of the backtracking algorithm from classical DTW [98]. By again applying the chain rule, we obtain $\mathbf{G}(n, m)$ as

$$\mathbf{G}(n, m) = \sum_{s=1}^{S+1} \frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{f}_s^{(n, m)}} \cdot \frac{\partial \mathbf{f}_s^{(n, m)}}{\partial \mathbf{C}(n, m)}, \quad (9.22)$$

where the first term in (9.22) is given by the backtracking matrix

$$\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{f}_s^{(n, m)}} = \mathbf{B}(n, m, s), \quad (9.23)$$

Figure 9.5: Musical score of the first measures of Beethoven’s piano sonata Op. 14 No. 2, first movement. The predominant theme is highlighted in red.

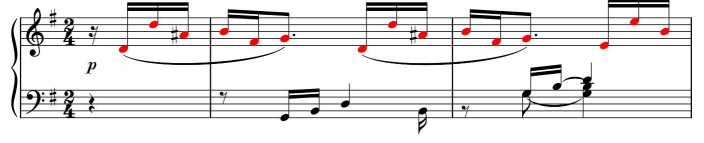
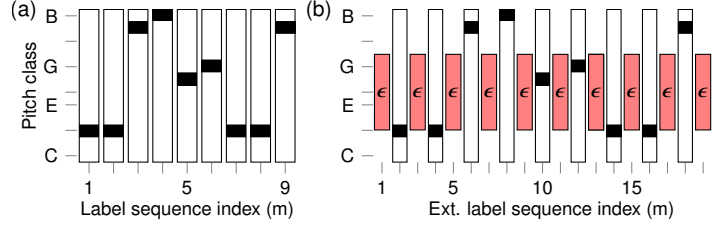


Figure 9.6: Single-class target representations for the pitch classes corresponding to the predominant theme of the running example in Figure 9.5 for the first two measures. (a) Label sequence Y . (b) Extended label sequence Y^e .



and the second term in (9.22) is given by either the step weight, or, in the case (n, m) is in the start boundary conditions, a factor of one:

$$\frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n,m)} = \begin{cases} \mathbf{W}(n, m, s), & \text{if } s \in [1 : S], \\ 1, & \text{if } s=S+1 \text{ and } (n, m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases} \quad (9.24)$$

The full backward pass is summarized in Algorithm 9.2. If we use unweighted dDTW, where all step weights are one ($\mathbf{W}$ is an all-one matrix), $\mathbf{G}$ also becomes an all-one matrix and thus $\mathbf{H} = \mathbf{E}$ [168]. The matrix $\mathbf{E}$ can be interpreted as a “soft alignment matrix” [30], with entries $\mathbf{E}(n, m)$ describing the probability of sequence elements x_n and y_m being aligned under the dDTW loss. We visualize the soft alignment matrix in Figure 9.11 and refer to Sections 4.3 and 6.2 and [30, 167, 168] for further explanations.

9.6 Label Probabilities and Cost Functions

In the previous sections, we analyzed CTC and SDTW from an alignment perspective, neither specifying how predictions X and targets Y are represented, nor how the probability $p(y_m^e | x_n)$ and local cost function $c(x_n, y_m)$ are defined. We now clarify these aspects using two concrete examples, one being a single-label (enhancement of predominant monophonic musical themes, see Section 9.7.2) and the other being a multi-label scenario (polyphonic pitch class estimation from music recordings, see Section 9.7.3). Both tasks, which are well-studied in Chapters 5 and 6 and in previous work [146, 160] serve three purposes: First, they illustrate how to define suitable label probabilities $p(y_m^e | x_n)$ for the single- and multi-label CTC loss. Second, they show how SDTW naturally accommodates even multi-label representations through appropriate cost functions, without increasing algorithmic complexity. Third, they clarify the conceptual and computational differences and similarities between CTC and SDTW.

9.6.1 Single-Label Classification

Estimating the activity of the twelve pitch classes $\{C, C\sharp, D, \dots, B\}$ (as typically used in 12-tone equal tempered scale in Western music) of a monophonic musical theme (see Figure 9.5) from an audio recording presents a single-label classification task. The estimated pitch classes can be used in applications such as theme-based music retrieval [160]. The extended alphabet A' contains the twelve pitch class symbols plus the blank label. We model these twelve targets plus the blank symbol as vectors with a one-hot encoding, i.e., $y_m^e \in \{0, 1\}^D$ (see Figure 9.6). $D \in \{12, 13\}$ is chosen depending on the requirement of an additional blank class. For example, in the score from Figure 9.5, we model the first note of the annotated theme (pitch class “D”) as a single-hot vector $y_1 = [0, 0, 1, 0, \dots, 0]$. Similarly, the network outputs vectors $x_n \in \Delta^D$, where Δ^D is the D -dimensional probability simplex, describing the predicted probabilities for all classes.

CTC

For CTC, we set $D = 13$ to include the blank label in the feature space. Each predicted vector x_n contains in its components the probabilities for all labels in the alphabet. Using vector notation, the label probabilities can be obtained as

$$p(y_m^e | x_n) = \langle x_n, y_m^e \rangle, \quad (9.25)$$

where the dot product $\langle \cdot, \cdot \rangle$ selects from the prediction vector x_n the probability for the label given by the one-hot encoding in y_m^e .

SDTW

For SDTW, we do not require the blank symbol (i.e., $D = 12$) and directly use the weak labels $Y = (y_1, \dots, y_M)$ as the target sequence. To mirror the CTC label probability specified in (9.25) with SDTW, we define the cost function as the negative log-likelihood for the true class label

$$c(x_n, y_m) = -\log \langle x_n, y_m \rangle \quad (9.26)$$

and, while disregarding the blank symbol, fulfill the requirement $c(x_n, y_m) = -\log p(y_m^e | x_n)$ given by (9.9). To avoid changing the network architecture, we model rests as all-zero vectors.

9.6.2 Multi-Label Classification

We now consider the joint estimation of pitch classes from recordings containing voice and piano, forming a polyphonic multi-label problem. As a scenario, we use songs for male voice and piano from Schubert’s “Winterreise”, contained in the multi-modal SWD [153]. The musical score (see

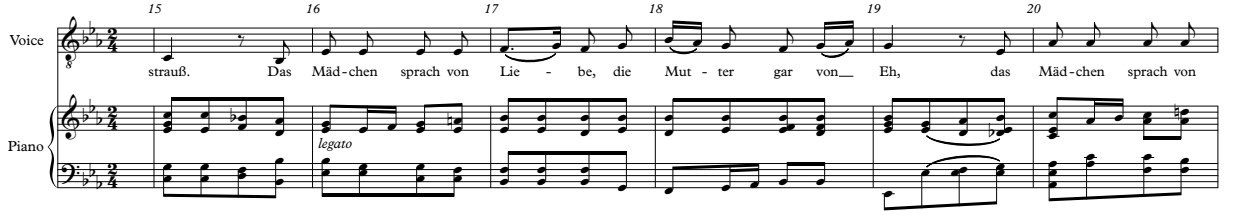


Figure 9.7: Excerpt from “Gute Nacht” (Schubert’s “Winterreise”), used as a running example for multi-label classification.

Figure 9.8: Multi-class target representation for the pitch classes of the running example in Figure 9.7 for measures 15 and 16. (a) Label sequence Y . (b) Extended label sequence Y^e .

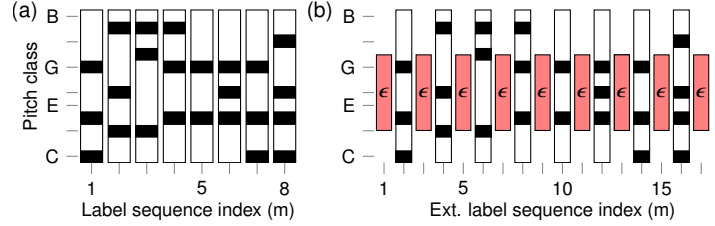


Figure 9.7 for an example) provides symbolic labels that are weakly aligned to the audio. The multi-class prediction of the simultaneously occurring pitch classes in an audio recording can be used, e.g., for subsequent harmony analysis [154] or as features for music synchronization [44]. We represent the multi-class labels $Y = (y_1, \dots, y_M)$ as twelve-dimensional multi-hot feature vectors $y_m \in \{0, 1\}^{12}$, encoding in each of the twelve dimensions the activity of the respective pitch class. For instance, the first chord in measure 15 contains the pitch classes $\{C, E^b, G\}$ and is encoded as a multi-hot vector $y_1 = [1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]$ (see Figure 9.8a). Predictions are represented similarly as $x_n \in [0, 1]^D$, where each component indicates the activation probability of a pitch class (and the blank label, if required, by choosing $D \in \{12, 13\}$).

Multi-Label CTC

In the original CTC formulation, all possible labels must be drawn from an alphabet A . For a multi-label task with 12 pitch classes, this implies $|A| = 2^{12} = 4096$ possible label combinations. Computing probabilities for all such combinations is infeasible in practice. Therefore, prior work on the Multi-Label Connectionist Temporal Classification (MCTC) [146, 156] restricts the alphabet to only those combinations actually occurring in the label sequence Y . In our setup, we directly define the conditional probabilities $p(y_m^e | x_n)$ for the elements of the extended label sequence y_m^e (see Figure 9.8b), which can be used to compute the CTC loss as in (9.9). Two cases must be distinguished based on the index m : If m is odd, $y_m^e = \epsilon$ (blank symbol), and if m is even, $y_m^e = y_{m/2}$ (the corresponding multi-hot label from Y), as defined in Section 9.2.1. Following [146, 156], we define multi-class probabilities

$$p(y_m^e | x_n) = \begin{cases} p(\epsilon | x_n), & \text{if } m \text{ is odd (blank),} \\ (p(\bar{\epsilon} | x_n) \cdot p(y_{m/2} | x_n)), & \text{else (not blank),} \end{cases} \quad (9.27)$$

where $p(\epsilon|x_n)$ and $p(\bar{\epsilon}|x_n)$ are the probabilities of presence and absence of the blank symbol, respectively, and $p(y_{m/2}|x_n)$ is the probability of the non-blank multi-hot label $y_{m/2} = y_m^e$. The encoding of predictions is identical to single-label CTC where we set $D = 13$, to extend each prediction by one dimension to include a probability for blank. Thus, we have $x_n \in [0, 1]^{13}$, where $x_{n,0}$ corresponds to the blank and $x_{n,i}$ (with $i \geq 1$) to pitch class activations. For the probabilities of the blank symbol, we then have:

$$p(\epsilon|x_n) = x_{n,0}, \quad p(\bar{\epsilon}|x_n) = 1 - x_{n,0}. \quad (9.28)$$

For a given multi-hot label $y_m \in \{0, 1\}^{12}$ and prediction x_n , the probability of y_m given x_n is modeled as the product over all pitch classes:

$$p(y_m|x_n) = \prod_{i=1}^{12} \left(y_{m,i} x_{n,i} + (1 - y_{m,i})(1 - x_{n,i}) \right), \quad (9.29)$$

assuming conditional independence of pitch classes.

SDTW

For the SDTW trainings, we directly use the weak labels $Y = (y_1, \dots, y_M)$ encoded as multi-hot vectors $y_m \in \{0, 1\}^{12}$ for training. Analogously, the predictions $X = (x_1, \dots, x_N)$ are considered probabilities for the pitch classes with $x_n \in [0, 1]^{12}$. We use the BCE as local cost function:

$$c_{\text{BCE}}(x_n, y_m) = - \sum_{i=1}^{12} \left(y_{m,i} \log x_{n,i} + (1 - y_{m,i}) \log(1 - x_{n,i}) \right), \quad (9.30)$$

which is a common choice in multi-label classification tasks.

9.6.3 Discussion

In both the single- and the multi-label configurations above, we observe equality in the cost values $c_{\text{BCE}}(x_n, y_m) = -\log p(y_m|x_n)$, aligning with the requirement formulated in (9.9). Although the CTC and SDTW objectives appear similar in form, they differ conceptually: First, CTC requires explicit modeling of a blank symbol, which impacts both the network architecture (adding an output dimension) and the alignment computation. Second, MCTC increases implementation complexity by constructing the probabilities of multi-class labels from component-wise single-class probabilities. In contrast, SDTW operates directly on two feature sequences using a local cost function, without the need for a blank label or probabilistic interpretation. This makes SDTW not only more intuitive and didactically accessible but also more flexible. The cost function c can be freely adapted (e.g., beyond BCE) without violating the underlying alignment assumptions, and SDTW naturally supports real-valued target sequences beyond binary multi-hot representations.

Table 9.1: Overview of different CTC and SDTW configurations within the dDTW framework. For explanations, see Section 9.7.1.

ID	has	(1,0)		(1,1)		(1,2)		(0,1)
	ϵ	tgt	ϵ	tgt	ϵ	tgt	ϵ	tgt
CTC-A	✓	1.0	1.0	1.0	1.0	1.0	∞	–
CTC-B	✓	1.0	2.0	1.0	2.0	1.0	∞	–
CTC-C	✓	1.0	∞	∞	∞	1.0	∞	–
SDTW-D	×	1.0	–	1.0	–	–	–	–
SDTW-E	×	1.0	–	1.0	–	–	–	1.0
CTC-A-W	✓	0.1	1.0	1.0	1.0	1.0	∞	–
CTC-B-W	✓	0.1	2.0	1.0	2.0	1.0	∞	–
CTC-C-W	✓	0.1	∞	∞	∞	1.0	∞	–
SDTW-D-W	×	0.1	–	1.0	–	–	–	–
SDTW-E-W	×	0.1	–	1.0	–	–	–	1.0

9.7 Experiments

In this section, we examine the core conceptual similarities and differences between CTC and SDTW. The proposed dDTW framework offers a unified view of both losses, allowing us to compare their behavior in a common language. With suitable parameter choices (Section 9.7.1), dDTW spans a continuum from SDTW to CTC, so we can study not only the two endpoints but also the effects of individual algorithmic components. Although dDTW enables task-specific tuning of step sizes and weights, softmax temperature, and even the minimum operator itself, a comprehensive ablation is beyond our scope. In the following, we focus on the parameter settings that recover CTC and SDTW and on representative configurations between them.

We start with a single-label task, predominant musical theme enhancement (Section 9.7.2), and evaluate frame-level accuracy and downstream musical theme retrieval. We then analyze sequence decoding and symbol-level evaluation: analogous to ASR, we decode predictions using greedy and beam search methods and measure symbol-level edit distances. Next, we consider multi-class settings (Section 9.7.3) in a polyphonic pitch-class estimation problem using both MCTC and SDTW, illustrating how SDTW extends to multi-label scenarios. Finally, in Section 9.7.4 we compare the computational cost in practice of classical SDTW and CTC implementations with our proposed dDTW framework.

Across all experiments, our goal is not to rank CTC against SDTW, but to use the dDTW continuum as an analytic tool. By moving through this continuum, we isolate specific components, most notably the CTC blank symbol, and study their effects on alignment, learning dynamics, and decoding behavior. For example, the blank is essential for sequence decoding and for separating repeated onsets, yet it is not needed to stabilize alignment in feature-learning tasks, where the same role can be achieved by a suitably parameterized SDTW variant. Code to reproduce all experiments is available at github.com/groupmm/dDTW-CTC.

9.7.1 Training Configurations

We now discuss the parameterization of various loss functions within our proposed dDTW framework. Table 9.1 provides an overview of the dDTW parameterizations leading to different training configurations. In Table 9.1, we distinguish alignment steps in a cell corresponding to a target y_m (tgt) or the blank symbol (ϵ). In the following section, we detail how dDTW generalizes both CTC and SDTW through appropriate parameter choices.

CTC

For all CTC configurations, we adopt the parameterization from Section 9.6. Specifically, we use the extended label sequence Y^e as targets, predictions X with $x_n \in [0, 1]^D$ with $D = 13$, and the local cost function $c(x_n, y_m^e)$ as defined in (9.25) and (9.27). We set dDTW step sizes to $\mathcal{S} = \{(1, 0), (1, 1), (1, 2)\}$ and prohibit skipping (step $(1, 2)$) of blank symbols by assigning infinite weight to corresponding entries in the weight matrix. Boundary conditions are set to $\mathcal{B}_{\text{start}} = \{(1, 1), (1, 2)\}$ and $\mathcal{B}_{\text{end}} = \{(N, M^e - 1), (N, M^e)\}$. We denote the baseline configuration with uniform step weights as CTC-A. Introducing a slight penalty for blank transitions by setting the step weights into and within blank symbols to two yields CTC-B. To obtain a configuration resembling SDTW, we disable all blank-related transitions by assigning infinite weights to the respective entries in the weight matrix, yielding CTC-C. If a transition through the blank symbol is required to distinguish repeated target symbols, we assign the weight of the $(1, 1)$ -step to that particular transition. Note that CTC-C retains blank labels in both the network and the weak targets, although the blank target is never aligned during loss computation. An illustration of the step sizes used for CTC parameterizations of dDTW is shown in Figure 9.4.

SDTW

For all SDTW-based configurations (SDTW), weak targets are given by the label sequence Y , and predictions X consist of twelve-dimensional vectors $x_n \in [0, 1]^D$ with $D = 12$. As detailed in Section 9.6, the local cost function is given by (9.26) and (9.30). We first parameterize an SDTW configuration SDTW-D that is similar to CTC in the sense that vertical alignment steps are not allowed, by choosing step weights $\mathcal{S} = \{(1, 0), (1, 1)\}$ and boundary conditions $\mathcal{B}_{\text{start}} = \{(1, 1)\}$ and $\mathcal{B}_{\text{end}} = \{(N, M)\}$. To parameterize the standard SDTW algorithm (SDTW-E) within the dDTW framework, we use step sizes $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$ with the same boundary conditions as SDTW-D.

Ablation Study for Horizontal Step Weight

Prior work on SDTW has shown that the weight w_h assigned to the horizontal step $(1, 0)$, which encodes the repetition of a target, critically affects training stability (see Chapter 6 and [168]). Specifically, reducing

Table 9.2: CNN architecture for musical theme enhancement from [160]. $Z \in \{0, 1\}$ denotes an optional output dimension for blank.

Layer	Kernel size	Output shape	# Parameters
Input		$(T, 216, 6)$	
Conv2D	3×3	$(T, 216, 64)$	3520
Conv2D	3×3	$(T, 216, 32)$	18464
Conv2D	3×3	$(T, 216, 32)$	9248
Conv2D	3×42	$(T, 216, 8)$	32264
Conv2D	1×1	$(T, 216, 1)$	9
Pooling		$(T, 12 + Z)$	$Z \cdot 217$
Total			72753 + $Z \cdot 217$

this weight improves alignment quality. To systematically investigate this, we run all experiments with either the standard horizontal step weight of $w_h = 1.0$ or a reduced weight of $w_h = 0.1$ (suffix -W).

Baselines

As a baseline, we use an Expectation–Maximization (EM)-like approach inspired by [90, 121], where, in every training step, we do hard DTW alignment between the weak target sequence (uniformly stretched to the length of the predictions), and the predicted sequence. The aligned target sequence is then used as element-wise targets for training, and the approach is denoted as EM. Furthermore, we train all networks on strongly aligned reference annotations (strong).

9.7.2 Single-Label Classification: Musical Theme Enhancement

As a first scenario, we consider a single-label classification task: enhancing pitch classes of monophonic predominant themes in audio recordings [160]. We use the Musical Theme Dataset (MTD) [162], which offers symbolic annotations for 2067 themes from 1126 classical pieces (about 5 h of annotated excerpts and 120 h of full recordings). Following Zalkow and Müller [160], we train a small CNN to produce a chromagram-like representation that highlights the pitch classes of the predominant theme. Each training example pairs an audio segment containing a theme with the corresponding pitch-class sequence extracted from the score (see Figures 9.5 and 9.6).

The setup is closely related to ASR training, where utterances are paired with unaligned label sequences; accordingly, [160] optimize the network using the CTC loss. Unlike ASR, they do not decode a transcription but use posterior probabilities directly as mid-level features for matching against prototype themes. We revisit this task as a compact testbed for our unified dDTW framework, allowing us to probe the continuum between CTC- and SDTW-like parameterizations and their effect on learning, both at the frame level (pitch-class accuracy and downstream retrieval) and at the symbol level (greedy/beam decoding and normalized edit distance to the reference themes).

Table 9.3: Experimental results for monophonic pitch-class enhancement based on MTD [162]. The table shows retrieval results for different loss configurations, reported as top- k rank and Mean Reciprocal Rank (MRR). For accuracy, we provide the mean value over five runs and the best individual run; for retrieval, we use the model with the lowest validation loss.

ID	Accuracy $\uparrow$		Retrieval rank $\uparrow$			MRR $\uparrow$
	mean	best	top 1	top 10	top 50	
CTC-A	0.473	0.574	0.870	0.939	0.963	0.895
CTC-B	0.555	0.583	0.832	0.923	0.952	0.864
CTC-C	0.588	0.590	0.795	0.894	0.941	0.830
SDTW-D	0.657	0.661	0.856	0.936	0.957	0.889
SDTW-E	0.648	0.654	0.870	0.939	0.960	0.893
CTC-A-W	0.230	0.230	0.000	0.000	0.048	0.005
CTC-B-W	0.228	0.230	0.000	0.005	0.082	0.007
CTC-C-W	0.311	0.646	0.843	0.926	0.963	0.871
SDTW-D-W	0.650	0.655	0.891	0.944	0.957	0.908
SDTW-E-W	0.643	0.650	0.864	0.934	0.963	0.889
EM	0.653	0.657	0.888	0.949	0.973	0.907
strong	0.676	0.679	0.878	0.947	0.968	0.901

Experimental Setup

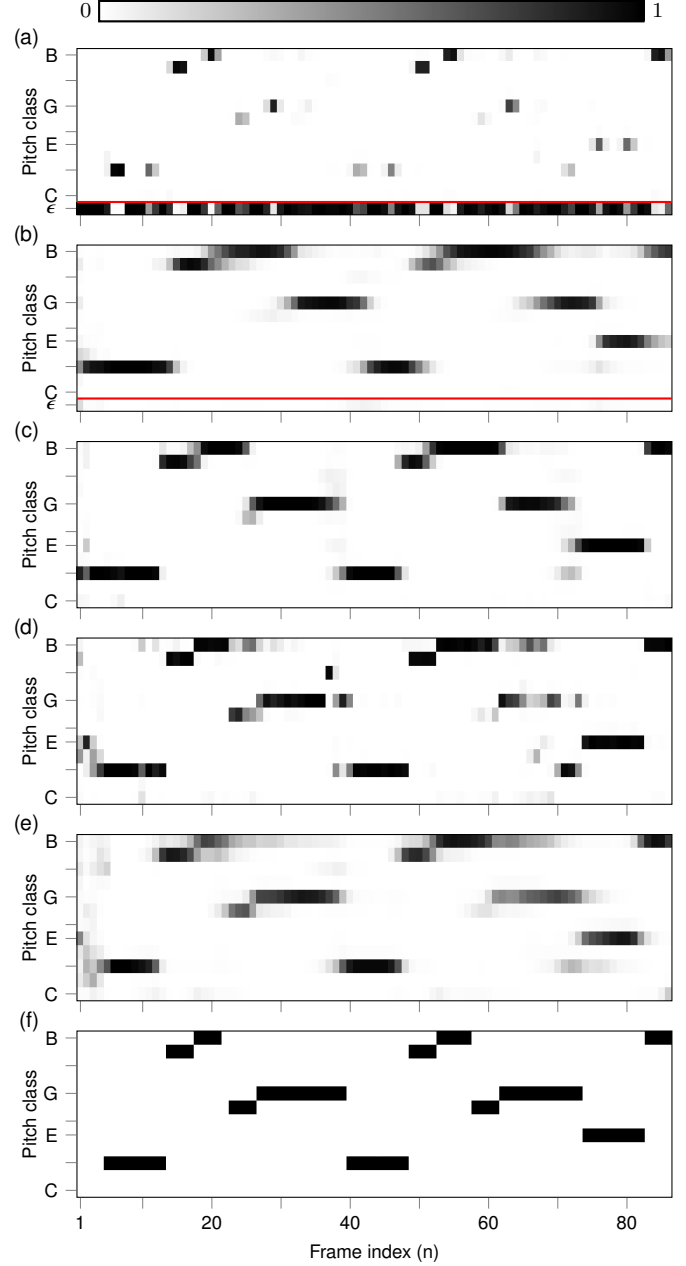
Throughout all experiments, we follow the training setup of [160] and vary only the loss within our dDTW framework. The CNN takes as input a HCQT [14] computed from audio sampled at 22 050 Hz with a hop size of 896 samples, corresponding to a frame rate of about 25 Hz. The HCQT spans six octaves with three bins per semitone (216 pitch bins) and six harmonic channels (one subharmonic plus five harmonics). Inputs are HCQT sequences of length T ; for each frame, the network outputs $x_n \in [0, 1]^D$, where $D \in \{12, 13\}$ indicates whether a blank symbol is included. The architecture is summarized in Table 9.2 and consists of 2D convolutional layers with leaky ReLU activations and a partly trainable pooling stage that reduces the output to D classes (see [160] for details). We train with minibatches of eight using Adam [74] at an initial learning rate of 0.001; the rate is halved if the validation loss does not improve for five epochs, and training stops if there is no improvement for ten epochs, restoring the model with the lowest validation loss.

Frame-Level Analysis

We assess monophonic theme predictions at the frame level using two measures: frame-wise pitch-class accuracy and retrieval performance based on the learned features. Frame-wise accuracy is the proportion of frames for which the most probable pitch class matches the strongly aligned reference. For retrieval, we process each database recording with the trained model to obtain an enhanced pitch-class representation and match it to a symbolic query theme via subDTW [98]. Rankings follow the minimum subDTW distance per query–document pair; we report top- k retrieval rates and Mean Reciprocal Rank (MRR) [160]. Results are summarized in Table 9.3 with example predictions in Figure 9.9.

Starting from standard CTC (CTC-A), we gradually relax constraints toward standard SDTW (SDTW-E). CTC-A yields low mean accuracy (0.473), dominated by blank activations (Figure 9.9a), yet achieves strong retrieval (top-1 = 0.870). Penalizing blanks (CTC-B) or forbidding blank alignments (CTC-C) increases

Figure 9.9: Predicted pitch classes for the musical theme visualized in Figure 9.5 for different network configurations. (a) CTC-A. (b) CTC-C. (c) SDTW-E. (d) EM. (e) *strong*. (f) Reference annotations.



accuracy to 0.555 and 0.588 and removes blank predictions (Figure 9.9b), but reduces retrieval, likely due to temporal blurring. SDTW-based losses substantially improve accuracy (SDTW-D 0.657, SDTW-E 0.648) and produce cleaner theme representations (Figure 9.9c), with top-1 retrieval of 0.856 and 0.870, respectively.

Lowering the horizontal step weight to 0.1 (suffix -W) severely degrades CTC variants and often prevents convergence, while SDTW variants remain largely stable. Although not best in accuracy, SDTW-D-W gives the strongest retrieval overall (top-1 = 0.891). The EM baseline performs consistently well in accuracy (0.653) and ranks second in retrieval. Training with strongly aligned targets (*strong*) yields the highest

Table 9.4: Experimental results for symbol-level decoding on MTD [162]. The table shows symbol-level Label Error Rate (LER) for different configurations of the DDTW loss. We report the mean metrics over all five training runs, as well as the metric for the respective best-scoring training run.

ID	LER (greedy) ↓		LER (beam search) ↓	
	mean	best	mean	best
CTC-A	0.282	0.212	0.256	0.189
CTC-B	0.251	0.220	0.431	0.207
CTC-C	0.271	0.257	0.618	0.594
SDTW-D	0.281	0.275	0.437	0.428
SDTW-E	0.285	0.266	0.415	0.400
CTC-A-W	0.500	0.500	1.999	1.997
CTC-B-W	0.476	0.470	1.793	1.288
CTC-C-W	0.433	0.276	1.893	0.955
SDTW-D-W	0.282	0.270	0.938	0.925
SDTW-E-W	0.278	0.271	1.013	0.967
EM	0.577	0.550	0.691	0.663
strong	0.263	0.256	0.752	0.724

frame-wise accuracy (0.676), but shows temporal blurring (Figure 9.9e), which likely explains slightly weaker retrieval than the best SDTW models.

Symbol-Level Decoding

Next, we apply CTC-style decoding to the predicted symbol posteriors. Greedy decoding selects the most probable symbol (including blank) per frame, collapses repeats, and removes blanks. Beam search [57] propagates the top- k collapsed sequences over time, sums probabilities of paths that collapse to the same sequence, and returns the most probable sequence. We evaluate both decoders using the normalized Levenshtein edit distance [86] between decoded and reference sequences, reported as Label Error Rate (LER), analogous to ASR character error rate.

With horizontal step weight $w_h = 1.0$, CTC and SDTW variants yield similar greedy LERs; CTC-B is best on average (0.251), and the best runs are dominated by CTC variants, with CTC-A achieving the lowest LER (0.212). Lowering w_h degrades CTC decoding, while SDTW models remain largely stable. Despite strong frame-level accuracy, EM produces substantially higher LER, likely because hard alignments introduce local outliers (Figure 9.9d) that barely affect frame accuracy but increase edit distance. The **strong** model outperforms all SDTW variants but remains slightly behind the best CTC settings, as only CTC with an explicit blank can represent consecutive repetitions (CTC-A, see Figure 9.9a), whereas SDTW and **strong** targets emphasize frame-wise activations rather than onsets (Figure 9.9c,e). Addressing this would require explicit onset modeling [63], which is beyond our scope.

Beam search shows a consistent trend: except for CTC-A, it increases LER, sometimes sharply, especially for reduced w_h where LER can approach or exceed 1.0. This suggests a mismatch between training-time path probabilities and beam-search sequence probabilities. In contrast, CTC-A, for which beam search was introduced [57], benefits from it, reducing mean LER to 0.256 and achieving the best-case LER of 0.189.

Table 9.5: CNN architecture for pitch class estimation from [146, 154]. $Z \in \{0, 1\}$ denotes an optional output dimension for blank.

Layer	Kernel size	Output shape	# Parameters
Input		$(T + 74, 216, 6)$	
Layer norm.		$(T + 74, 216, 6)$	2592
Conv2D, MP	15×15	$(T + 74, 216, 20)$	27020
Conv2D, MP	3×3	$(T + 74, 72, 20)$	3620
Conv2D	75×1	$(T, 72, 10)$	15010
Conv2D	1×1	$(T, 72, 1)$	11
Conv2D	1×61	$(T, 12 + Z)$	$62 + 73 \cdot Z$
Total			$48315 + 73 \cdot Z$

9.7.3 Multi-Label Classification: Polyphonic Pitch Class Estimation

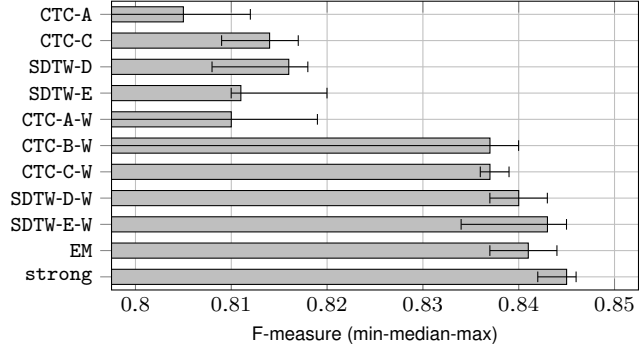
For our experiments on multi-label classification, we adopt a simple and well-established scenario: polyphonic pitch class estimation using CNNs [154]. Unlike larger music transcription models that require pretraining on strongly aligned data [63], this setup enables training from scratch using only weakly aligned data. To ensure comparability with prior work, we use the SWD dataset [153] and a musically motivated CNN architecture from [154]. This setup has previously been explored with strongly aligned targets [154], and with weakly aligned targets using MCTC [146] or SDTW (see Chapter 5 [167] and Chapter 6 [168]).

Experimental Setup

The CNN receives as input a HCQT [14], computed from audio at a sampling rate of 22 050 Hz with a hop size of 512 samples, resulting in a frame rate of approximately 43 Hz. The HCQT spans six octaves with three bins per semitone (resulting in 216 pitch bins) and includes six harmonic channels (one sub-harmonic and five harmonics). The input to the network consists of HCQT sequences with $T = 500$ frames and 74 additional context frames, where the segmentation is derived from strongly aligned reference annotations provided in the SWD. For each time frame, the network outputs a tensor $x_n \in [0, 1]^D$, where $D \in \{12, 13\}$ accounts for the presence of the blank symbol when required. A brief overview of the architecture, consisting of convolutional (Conv2D) and max pooling (MP) layers, is provided in Table 9.5. We refer to [146, 154] for the design motivation and further details.

The SWD contains approximately 11 h of paired scores and audio recordings by seven different singers and nine different pianists, resulting in nine distinct versions. We split the SWD into a training, validation, and test set with five, two, and two versions, respectively. Training is performed on single-element batches. Each epoch comprises approximately 4500 training steps, and we train for 50 epochs. We use the Adam optimizer [74] with an initial learning rate of 0.001, which is halved if the validation loss does not improve over 4 epochs. Training stops if there is no improvement over 12 epochs, and the model parameters corresponding to the epoch with the lowest validation loss are restored for testing.

Figure 9.10: Test results for the five training runs of each model configuration. The bar represents the median F-measure for frame activation over the five training runs, the whiskers represent the minimum and maximum F-measure of the training runs.



For qualitative illustration, we use a 10 s excerpt from the song “Gute Nacht” performed by Randall Scarlata, with the corresponding score shown in Figure 9.7 and strongly aligned reference targets shown in Figure 9.11f.

Analysis: Pitch Class Accuracy

In this section, we present systematic experiments enabling a detailed comparison of different CTC and SDTW configurations. Starting from the standard CTC-A, we progressively modify the parameterization until reaching the standard SDTW-E. For each configuration, Figure 9.10 reports the minimum, median, and maximum F-measure over five training runs. For selected, didactically relevant configurations, Figure 9.11 visualizes the predictions X and soft alignment matrix E of the model with the lowest validation loss, applied to the running example from Figure 9.7. For the CTC configurations, we color odd rows of the soft alignment matrix in red (corresponding to alignment of the blank symbol ϵ), and even rows in black (corresponding to alignment of target symbols y_m). We begin by parameterizing all algorithms with a horizontal $(1, 0)$ step weight of $w_h = 1.0$, corresponding to the standard configurations in both CTC and SDTW.

Starting with the original CTC algorithm (CTC-A), the test F-measure shown in Figure 9.10 is relatively low at 0.81, and shows high variance across different runs. The predictions in Figure 9.11a reveal a strong bias toward the blank symbol, along with large temporal fluctuations in the pitch class activations. This is reflected in the soft alignment matrix, where the dominance of red color indicates that the alignment largely focuses on the blank symbol.

To reduce this blank dependence, we introduce a penalty for blank alignment by increasing the step weight to 2 for steps in and to the blank symbol (CTC-B). However, none of the training runs converged to a stable solution under this setup. We hypothesize that this occurs because the network relies on the blank symbol to produce stable alignments (as seen in CTC-A), but the increased penalty now makes prolonged blank alignment prohibitively costly, while the blank symbol still remains dominant enough to distract from learning the actual targets.

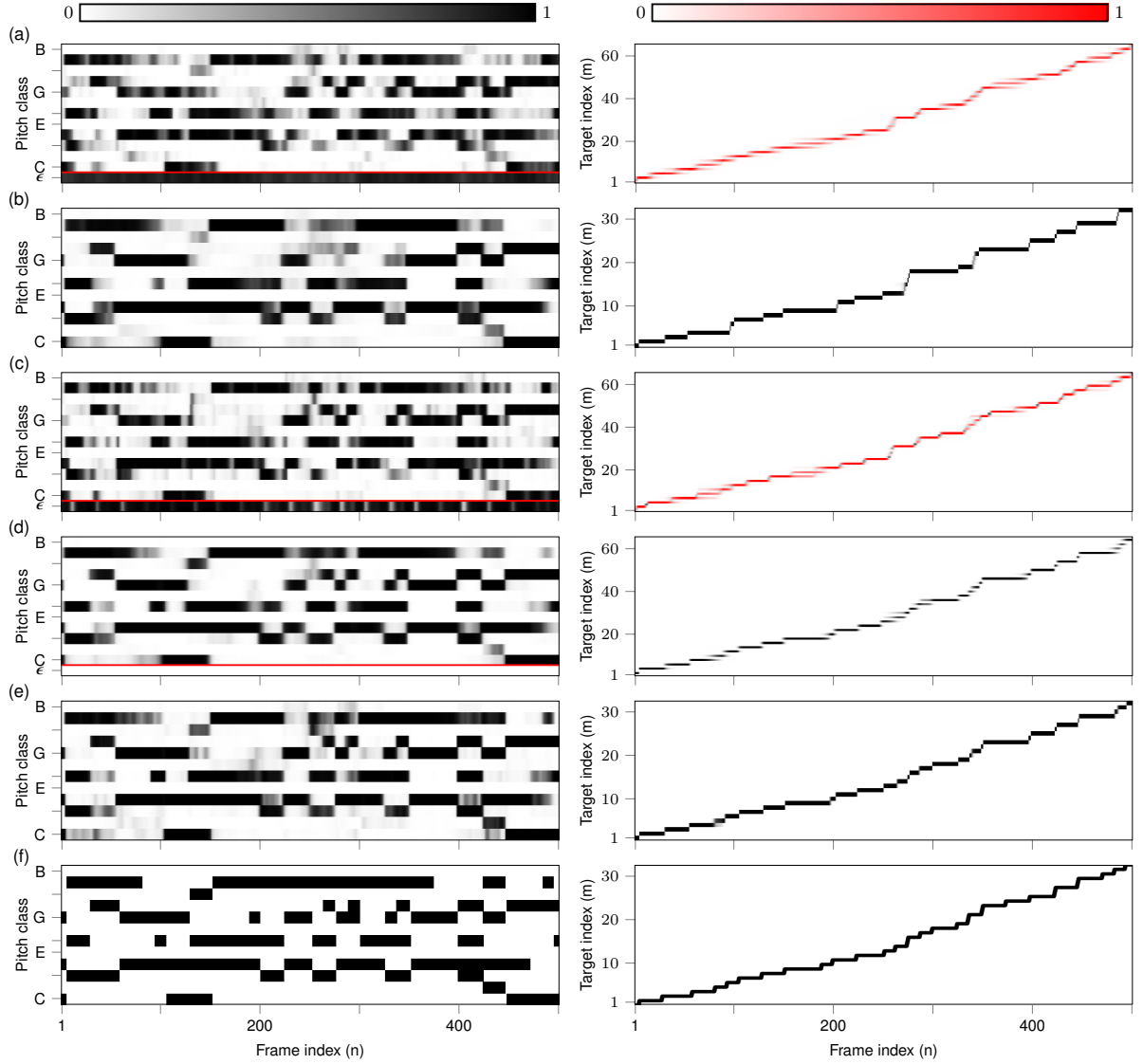


Figure 9.11: Predictions (left) and soft alignments (right) for different network configurations for the running example. (a) CTC-A. (b) SDTW-E. (c) CTC-A-W. (d) CTC-B-W. (e) SDTW-E-W. (f) Strongly aligned labels and alignment obtained from reference annotations available in the SWD. Red color is used for the blank symbol ϵ .

Next, we impose the largest possible penalty by setting an infinite weight for steps in and to the blank symbol (CTC-C). According to Figure 9.10, this leads to a notable improvement in F-measure compared to CTC-A, with low variance across runs. Because blank alignments are effectively forbidden, only actual target labels are aligned during training. This setup resembles SDTW in the sense that no blank symbol is aligned, although the blank dimension remains present in the network outputs.

We then train with SDTW without vertical step (SDTW-D), which is identical to CTC-C except that the blank symbol is completely removed from both the network output and the target sequence. Test performance in Figure 9.10 is very similar but slightly better on average than CTC-C. This improvement likely stems from the more efficient training due to the absence of the blank symbol in the output space.

Finally, we include the vertical step and evaluate the standard SDTW parameterization (SDTW-E). Here, the median test F-measure is slightly lower than SDTW-D, but the minimum and maximum over all runs are comparable. The predictions in Figure 9.11b show temporally stable but slightly blurred outputs. The soft alignment matrix reveals some alignment instabilities compared to the reference alignment in Figure 9.11f, with certain targets being aligned over extended durations, while others are assigned to only short intervals. We address this issue next by lowering the horizontal step weight to enforce more stable target repetitions.

Ablation: Sensitivity w.r.t. Horizontal Step Weight

Prior experiments on SDTW shows that lowering the horizontal step weight w_h stabilizes training and improves alignments (see Chapter 6 and [168]). High w_h causes alignments to concentrate on a few targets that minimize total cost, while a low w_h allows the alignment to remain on a target despite imperfect predictions, e.g., during sustained notes. We thus again repeat the previous experiments with setting weight $w_h = 0.1$ to the horizontal $(1, 0)$ step.

For the original CTC (CTC-A-W), Figure 9.10a shows an improvement in median and maximum F-measure versus CTC-A, though some runs do not converge to a good solution. Predictions in Figure 9.11c reveal slightly reduced probabilities for the blank symbol compared to CTC-A (Figure 9.11a), but only for short time intervals. The soft alignment matrix still shows a dominant alignment of the blank symbol (red color), although target symbols (black color) are aligned slightly more compared to CTC-A.

Similarly, lowering w_h while penalizing the blank symbol (CTC-B-W) improves median and maximum F-measures considerably (Figure 9.10), yet convergence issues remain. This suggests that combining two stabilization strategies (presence of blank symbol and low horizontal step weight) can destabilize training. Predictions in Figure 9.11d show low probabilities for the blank symbol, resembling SDTW-E, and the soft alignment matrix does not show alignment of the blank symbol anymore.

Conversely, excluding the blank symbol entirely (CTC-C-W) yields almost no F-measure variability across runs, matching results for SDTW without vertical step (SDTW-D-W).

Including the vertical step gives standard SDTW with low horizontal step weight (SDTW-E-W). Although the variance of the F-measure in Figure 9.10 is slightly wider than for SDTW-D-W, peak performance exceeds that of the models trained with an EM approach (EM), and nearly matches that of models trained on strongly aligned references (**strong**), with an F-measure of over 0.84. Predictions in Figure 9.11e show sharp, temporally stable outputs with minimal fluctuations, and the soft alignment matrix resembles the reference alignment to a high degree.

Table 9.6: Overview of runtime (milliseconds) and peak memory consumption (megabytes) for the SDTW, dDTW, and CTC loss functions for different sequence lengths (N, M) .

(N, M)	SDTW		dDTW		CTC	
	Time	Mem.	Time	Mem.	Time	Mem.
(256, 64)	3.9	73	8.3	23	1.2	1
(256, 128)	6.8	146	8.8	45	0.8	2
(512, 64)	7.3	146	7.3	46	1.1	3
(512, 128)	12.4	293	10.1	91	1.1	3
(1024, 128)	20.4	586	12.1	182	1.4	7
(1024, 512)	61.9	2345	26.6	722	1.4	8
(1024, 1024)	118.5	4689	43.0	1441	1.4	8

9.7.4 Runtime and Memory Consumption

We analyze runtime and memory usage of three alignment-loss implementations: standard SDTW¹³ [88], our dDTW toolbox¹⁴, and the optimized C++ PyTorch CTC loss¹⁵ [115]. Our goal is not to describe implementation internals but to estimate computational costs in typical DNN training. All experiments were conducted on an Nvidia RTX A5500 GPU.

For SDTW and dDTW, we initialize predictions and labels as $\mathbf{X} \in [0, 1]^{B \times N \times D}$ and $\mathbf{Y} \in \{0, 1\}^{B \times M \times D}$; for CTC, labels are $\mathbf{Y}_{\text{CTC}} \in [1 : D]^{B \times M}$. We choose a batch size of $B = 16$ and $D = 12$ feature dimensions. Each iteration randomly initializes inputs, performs a forward–backward pass, and we record mean runtime and peak memory consumption over 10 warm-up and 20 measured iterations. Both SDTW and dDTW have $O(NM)$ recursions but can be parallelized along anti-diagonals [88, 172], yielding $O(N + M)$ parallel time. The strictly monotonic CTC dynamic program runs in $O(N)$ parallel steps. All methods parallelize over the batch, though only dDTW and CTC support variable-length sequences within a batch.

Table 9.6 summarizes results. dDTW uses roughly one third of the memory of SDTW, benefiting from more efficient memory management and explicitly defined cost functions without autograd leakage. For short sequences, both perform similarly in speed; for longer sequences, dDTW becomes substantially faster than the baseline SDTW implementation, likely due to more effective caching (including the backtracking tensor $\mathbf{B}$). PyTorch’s native CTC is by far the fastest and most memory-efficient: runtimes stay near one millisecond and memory usage is about 180× lower than dDTW and 580× lower than SDTW for long sequences. While our research-oriented dDTW toolbox prioritizes modularity and clarity, future work may bring dDTW efficiency closer to that of the highly optimized CTC implementation.

9.7.5 Summary

Our experiments show that while both CTC and SDTW can effectively train neural networks, they differ in several theoretical and practical aspects. Importantly, the dDTW loss unifies both alignment paradigms

¹³ <https://github.com/Maghoumi/pytorch-softdtw-cuda>

¹⁴ https://github.com/groupmm/dDTW_CTC

¹⁵ <https://github.com/pytorch/pytorch>

within a single framework, enabling fine-grained control of loss parameters and providing a continuous interpolation between the two conceptually different losses. In the following, we highlight three key differences observed in our experiments.

Network Architecture

CTC requires an explicit output unit for the blank label. As a result, models pre-trained on strongly aligned data cannot be directly fine-tuned with CTC, since their output space does not include a blank. SDTW imposes no such architectural requirement and can be applied to any model without modification.

Target Length and Runtime

Because CTC inserts a blank symbol between all labels, the effective target length doubles compared to SDTW without blanks. Under a naïve $O(NM)$ implementation, this would double runtime. In practice, GPU-parallel implementations of SDTW [88] and dDTW iterate over anti-diagonals of the cost matrix. Since $N \gg M$ in typical settings, runtime is dominated by N , making the theoretical runtime of CTC comparable to that of SDTW. However, CTC is the de facto standard for training with unaligned sequential data, and highly optimized C++/CUDA implementations exist [115], which outperform research-oriented SDTW and dDTW implementations by a wide margin.

Detecting Repetitions

In CTC, the blank symbol disambiguates consecutive identical labels (symbol–blank–symbol). While irrelevant for frame-wise pitch-class activity estimation, this mechanism is essential for onset-sensitive tasks such as sequence estimation in theme enhancement, instrument onset detection, and ASR. In SDTW, this functionality has to be explicitly replicated by introducing a dedicated onset class, enabling joint estimation of onsets and frame activations, as demonstrated in models using the Onset-and-Frames architecture [63, 77].

9.8 Further Notes

In this chapter, we established a theoretical connection between the CTC and SDTW alignment paradigms. By identifying necessary extensions for SDTW to exactly replicate CTC in practice, we introduced a Differentiable Dynamic Time Warping (dDTW) framework, providing a unified perspective on differentiable alignment methods. We derived efficient DP recursions for dDTW and demonstrated how specific parameter settings recover standard CTC and SDTW. Leveraging this unified framework, we systematically analyzed the impact of the blank symbol, step sizes, and step weights in two controlled

experimental scenarios. Our results indicate that a carefully chosen SDTW parameterization, specifically, with a low horizontal step weight, renders the blank symbol unnecessary for alignment stability and improves prediction performance for frame-wise metrics. However, for symbol-level decoding, especially in the presence of repeated symbols, the CTC yields predictions with a lower error rate than SDTW without the blank symbol. Future work may explore modelling repeated symbols in dDTW without relying on a blank symbol by, e.g., using onset models.

10 A Graph Perspective on Differentiable Alignment

All alignment methods introduced so far share a common computational structure that can be expressed as a shortest-path problem on weighted Directed Acyclic Graphs (DAGs). Making this structure explicit clarifies which design choices are essential, which are interchangeable, and under what conditions the efficient DP recursions are valid.

This chapter formulates differentiable sequence alignment as a graph-based minimum-cost path finding problem. It derives the corresponding forward and backward recursions with correctness guarantees, naturally emerging from the graph-based path-finding perspective. We show how DTW, SDTW, CTC, and the proposed dDTW variants arise as special cases of a unified graph template. This abstraction provides formal validity guarantees for DP formulations and consolidates the technical framework of this thesis. Furthermore, it offers a basis for extending differentiable alignment to additional matching paradigms.

The remainder of this chapter is structured as follows. Section 10.1 introduces shortest-path problems on DAGs. Building on this foundation, Section 10.2 formulates sequence alignment as a shortest-path problem on highly structured DAGs. This structure allows for efficient DP-based path-finding algorithms, which are derived in Section 10.3. Next, Section 10.4 presents the explicit forward and backward passes of the dDTW loss function, as implemented in the dDTW toolbox. Section 10.5 outlines how a variety of sequence alignment paradigms can be expressed through specific parameterizations within the proposed graph-based dDTW framework. Section 10.6 concludes this chapter and connects the developed ideas to possible future research.

10.1 Graphs and Shortest-Path Problems

In optimization and machine learning, many problems can be formulated as shortest-path problems on graphs. A weighted directed graph is defined as $G = (V, E)$, where V denotes a set of vertices and $E \subseteq V \times V$ denotes a set of directed edges. Each edge $e = (v_i \rightarrow v_j) \in E$ that connects two vertices $v_i, v_j \in V$ is associated with a weight $\theta_e \in \mathbb{R}$, and we denote the set of all edge weights by $\Theta = \{\theta_e \mid e \in E\}$ [79].

A path $P = (p_1, \dots, p_L) \in V^L$ is a sequence of vertices such that each consecutive pair of vertices is connected by a directed edge, i.e.,

$$(p_{\ell-1} \rightarrow p_\ell) \in E \quad \text{for } \ell = 2, \dots, L. \quad (10.1)$$

The cost $C(P)$ of a path is defined as the sum of the weights of its edges:

$$C(P) = \sum_{\ell=2}^L \theta_{(p_{\ell-1} \rightarrow p_\ell)}. \quad (10.2)$$

Given a source vertex $p_{\text{start}} \in V$, the single-source shortest-path problem consists of finding the minimum-cost path from p_{start} to every other vertex $p \in V$. A classical method for solving this problem is *Dijkstra's algorithm* [34]. With an efficient implementation, the algorithm computes all shortest paths with time complexity $O(|E| + |V| \log |V|)$ [79].

If a directed graph is acyclic, i.e., it contains no directed cycles, the shortest-path problem becomes significantly simpler. In this case, the vertices admit a *topological ordering* $v_1, \dots, v_{|V|}$ such that every edge $(v_j \rightarrow v_i) \in E$ satisfies $j < i$. Processing the vertices in this order allows computing shortest-path distances to all vertices by visiting each vertex exactly once. Consequently, the shortest-path problem on a DAG can be solved in linear time with complexity

$$O(|V| + |E|). \quad (10.3)$$

In the following, we formulate sequence alignment as a shortest-path problem on a weighted DAG with a highly structured topology, which enables the use of particularly efficient dynamic programming algorithms.

10.2 Sequence Alignment as a Shortest-Path Problem on Graphs

In this section, we formulate the problem of aligning two sequences X and Y as a minimum-cost path problem on graphs, building on a theoretical framework introduced in [94]. In the following, we specify the graph in terms of vertices and directed edges with associated edge weights, as well as start and end conditions. An example for the dDTW graph is illustrated in Figure 10.1.

Vertices

Let $X = (x_1, \dots, x_N)$ and $Y = (y_1, \dots, y_M)$ denote two sequences with $x_n \in \mathcal{F}_X$ and $y_m \in \mathcal{F}_Y$ drawn from two suitable feature spaces. Furthermore, let $c : \mathcal{F}_X \times \mathcal{F}_Y \rightarrow \mathbb{R}$ be a local cost function that quantifies the

10.2. Sequence Alignment as a Shortest-Path Problem on Graphs

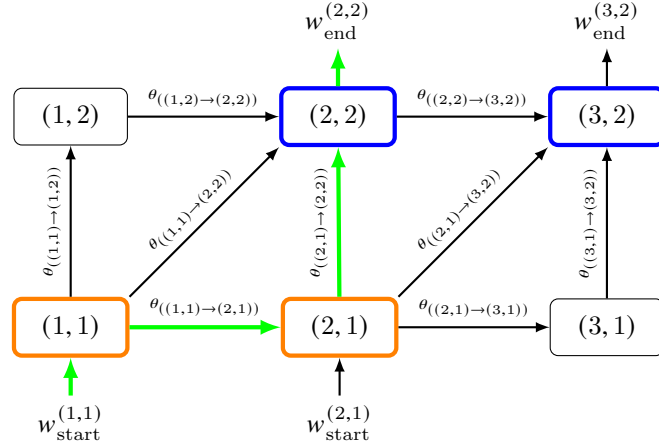


Figure 10.1: Example for a dDTW graph with vertices $\mathcal{I} = [1 : 3] \times [1 : 2]$, step sizes $\mathcal{S} = \{(1,0), (0,1), (1,1)\}$, edge weights $\theta_{((n',m') \rightarrow (n,m))}$, boundary conditions $\mathcal{B}_{\text{start}} = \{(1,1), (2,1)\}$ (orange) and $\mathcal{B}_{\text{end}} = \{(2,2), (3,2)\}$ (blue). A possible path $P = ((1,1), (2,1), (2,2)) \in \mathcal{P}$ is illustrated in green.

dissimilarity between two elements $x_n \in X$ and $y_m \in Y$. We represent all index pairs (n, m) of sequence elements (x_n, y_m) as vertices on a grid. While we denoted arbitrary vertices as V previously, the set of all vertices on the alignment grid is now given by the set:

$$\mathcal{I} := [1:N] \times [1:M]. \quad (10.4)$$

Edges

Directed edges $((n', m') \rightarrow (n, m))$ connect vertices (n', m') and (n, m) if their index difference corresponds to an allowed step from a set of step sizes

$$\mathcal{S} := \{(i_s, j_s) \mid s \in [1:S]\}, \quad (10.5)$$

where $(i_s, j_s) \in \mathbb{Z}_{\geq 0}^2$ must satisfy the (weak) monotonicity constraint $i_s + j_s \geq 1$ for all $(i_s, j_s) \in \mathcal{S}$. With this notation, (n', m') and (n, m) are adjacent if and only if $(n - n', m - m') \in \mathcal{S}$. The full set of directed edges E is then given by

$$E := \{((n', m') \rightarrow (n, m)) \mid (n - n', m - m') \in \mathcal{S}, (n', m') \in \mathcal{I}, (n, m) \in \mathcal{I}\}. \quad (10.6)$$

Since $i_s, j_s \geq 0$ and $i_s + j_s \geq 1$, each edge strictly increases at least one index, which prevents directed cycles. Thus, the graph $(\mathcal{I}, E)$ is a DAG.

Edge Weights

We define weights for all steps by a tensor $\mathbf{W} \in \mathbb{R}^{N \times M \times S}$, where $\mathbf{W}(n, m) \in \mathbb{R}^S$ contains, for each vertex (n, m) , the weights of all incoming steps defined by $\mathcal{S}$. Let $\theta_{((n', m') \rightarrow (n, m))}$ denote the cost of traversing from vertex (n', m') to (n, m) , i.e., the weight of the edge $((n', m') \rightarrow (n, m))$. This edge weight is given by the product of the local cost $c(x_n, y_m)$ and the corresponding step weight:

$$\theta_{((n', m') \rightarrow (n, m))} := c(x_n, y_m) \cdot \mathbf{W}(n, m, s), \quad (10.7)$$

with the step index s satisfying $(i_s, j_s) = (n - n', m - m') \in \mathcal{S}$. The set of all edge weights Θ is given by

$$\Theta := \{\theta_{((n', m') \rightarrow (n, m))} \mid ((n', m') \rightarrow (n, m)) \in E\}. \quad (10.8)$$

Boundary Conditions

We define the sets of vertices where paths can start and end as $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$, with associated start and end costs

$$w_{\text{start}}^{(n, m)} \in \mathbb{R} \quad \text{for all} \quad (n, m) \in \mathcal{B}_{\text{start}}, \quad (10.9)$$

$$w_{\text{end}}^{(n, m)} \in \mathbb{R} \quad \text{for all} \quad (n, m) \in \mathcal{B}_{\text{end}}. \quad (10.10)$$

Graphs and Paths

An alignment graph G is now defined as a weighted DAG over vertices $\mathcal{I}$, edges E with weights Θ , and start/end boundary conditions $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}}$ with their respective weights $w_{\text{start}}^{(n, m)}, w_{\text{end}}^{(n, m)}$ as given by (10.4) to (10.10):

$$G := (\mathcal{I}, E, \Theta, \mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}}, w_{\text{start}}^{(n, m)}, w_{\text{end}}^{(n, m)}). \quad (10.11)$$

A path $P = (p_1, \dots, p_L) \in \mathcal{I}^L$ over G is a sequence of vertices $p_\ell = (n_\ell, m_\ell)$ connected by edges $(p_{\ell-1} \rightarrow p_\ell) \in E$, subject to the boundary constraints $p_1 \in \mathcal{B}_{\text{start}}$ and $p_L \in \mathcal{B}_{\text{end}}$. The set of all paths over G is denoted by $\mathcal{P} := \{(p_1, \dots, p_L) \mid p_1 \in \mathcal{B}_{\text{start}}, p_L \in \mathcal{B}_{\text{end}}, (p_{\ell-1} \rightarrow p_\ell) \in E\}$, where the path length L can vary between different paths.

Minimum-Cost Path

The cost $C(P)$ of a path P is defined as the sum of the start, end, and edge costs

$$C(P) := w_{\text{start}}^{p_1} + \sum_{\ell=2}^L \theta_{(p_{\ell-1} \rightarrow p_\ell)} + w_{\text{end}}^{p_L} \quad (10.12)$$

10.3. Solving the Minimum-Cost Problem with Dynamic Programming

for $P = (p_1, \dots, p_L)$. The alignment cost $\text{dDTW}(X, Y)$ of X and Y is then defined as the minimum-cost path finding problem on G , taking into account the path costs $C(P)$:

$$\text{dDTW}(X, Y) := \mu \left((C(P))_{P \in \mathcal{P}} \right). \quad (10.13)$$

Throughout this chapter, the operator $\mu : \mathbb{R}^I \rightarrow \mathbb{R}$ denotes a differentiable minimum operator fulfilling certain properties (see Theorem B.1) defined for an arbitrary finite index set I . This index set may vary depending on the context, for example, in (10.13) we have $I = \mathcal{P}$.

Computational Complexity

When evaluating (10.13) directly, the cost $C(P)$ needs to be computed for all paths $P \in \mathcal{P}$, i.e., the naïve computational complexity of (10.13) is $\mathcal{O}(|\mathcal{P}|)$. The cardinality of the set $\mathcal{P}$ scales exponentially with the sequence lengths N and M and, for a single-source single-target problem with unit steps in the horizontal, diagonal, and vertical direction, can be calculated with the Delannoy formula $\mathcal{D}(N, M)$ [8] as:

$$|\mathcal{P}| = \mathcal{D}(N, M) = \sum_{k=0}^{\min(N, M)} \binom{N}{k} \binom{M}{k} 2^k. \quad (10.14)$$

However, because the dDTW graph is acyclic and highly structured, the minimum-cost path can be computed efficiently using DP, as detailed in the following section.

10.3 Solving the Minimum-Cost Problem with Dynamic Programming

In this section, we show how the minimum-cost problem from (10.13) can be solved using Dynamic Programming (DP) recursions. We base the derivation on previous results by Mensch and Blondel [94]. Let $\mathcal{P}(p) = \{(p_1, \dots, p_L) \mid p_1 \in \mathcal{B}_{\text{start}}, p_L = p, (p_{\ell-1} \rightarrow p_\ell) \in E\}$ define the set of all paths starting at a valid start vertex and ending in p . Let L denote the length of path $(p_1, \dots, p_L)$, which may vary across paths. Furthermore, let $\Psi(p) = \{q \in \mathcal{I} \mid (q \rightarrow p) \in E\}$ denote the set of all parent vertices to p , i.e., vertices from which an edge leads directly to p .

We define $\mathbf{D}(p)$ to be the minimum cost over all paths $\mathcal{P}(p)$ leading to p :

$$\mathbf{D}(p) = \mu \left(\left(w_{\text{start}}^{p_1} + \sum_{\ell=2}^L \theta_{(p_{\ell-1} \rightarrow p_\ell)} \right)_{(p_1, \dots, p_L) \in \mathcal{P}(p)} \right), \quad (10.15)$$

where the last vertex is $p_L = p$. Next, we modify (10.15) by removing the last edge $(p_{L-1} \rightarrow p_L)$ of each path from the sum:

$$\mathbf{D}(p) = \underbrace{\mu\left(w_{\text{start}}^{p_1} + \left(\sum_{\ell=2}^{L-1} \theta_{(p_{\ell-1} \rightarrow p_{\ell})}\right)\right)}_{\text{cost of paths ending in } \Psi(p)} + \theta_{(p_{L-1} \rightarrow p_L)} \Big|_{(p_1, \dots, p_L) \in \mathcal{P}(p)}. \quad (10.16)$$

Assuming associativity of the minimum operator, i.e., $\mu(a, b, c) = \mu(\mu(a, b), c)$, we split the problem into sub-tasks across the parent vertices $q \in \Psi(p)$ of p , where $p_L = p$ and $p_{L-1} = q$:

$$\mathbf{D}(p) = \mu\left(\mu\left(w_{\text{start}}^{p_1} + \left(\sum_{\ell=2}^{L-1} \theta_{(p_{\ell-1} \rightarrow p_{\ell})}\right) + \theta_{(p_{L-1} \rightarrow p_L)}\right) \Big|_{(p_1, \dots, p_{L-1}) \in \mathcal{P}(q)}\right)_{q \in \Psi(p)}. \quad (10.17)$$

Last, using the distributivity property of the minimum operator $\mu(a + c, b + c) = \mu(a, b) + c$, we extract the last edge weight $\theta_{(p_{L-1} \rightarrow p_L)} = \theta_{(q \rightarrow p)}$ from the sum:

$$\mathbf{D}(p) = \mu\left(\underbrace{\mu\left(w_{\text{start}}^{p_1} + \left(\sum_{\ell=2}^{L-1} \theta_{(p_{\ell-1} \rightarrow p_{\ell})}\right)\right) \Big|_{(p_1, \dots, p_{L-1}) \in \mathcal{P}(q)}}_{:=\mathbf{D}(q)} + \theta_{(q \rightarrow p)}\right)_{q \in \Psi(p)}, \quad (10.18)$$

where, by definition from (10.15), the inside expression for $\mu()$ can be substituted by $\mathbf{D}(q)$. Finally, this yields the DP recursion:

$$\mathbf{D}(p) = \mu\left(\mathbf{D}(q) + \theta_{(q \rightarrow p)}\right)_{q \in \Psi(p)}. \quad (10.19)$$

The recursion can be evaluated if the calculation of $\mathbf{D}(p)$ follows a linear ordering of nodes such that for every directed edge $(u \rightarrow v)$, u appears before v in the recursion. The final minimum cost value can now be trivially obtained by taking the minimum over the end vertices:

$$\text{dDTW}^{\text{DP}}(X, Y) = \mu\left(\mathbf{D}(p) + w_{\text{end}}^p\right)_{p \in \mathcal{B}_{\text{end}}}. \quad (10.20)$$

Thus, as long as μ satisfies associativity and distributivity criteria (which is the case for the hard minimum function or approximations with entropic regularization, i.e., softmin, see Section B.2), the global solution is identical to the DP result:

$$\text{dDTW}(X, Y) = \text{dDTW}^{\text{DP}}(X, Y). \quad (10.21)$$

10.4 Practical Computation of dDTW Cost and Gradient

In this section, we describe the practical approach for calculating the dDTW cost $\text{dDTW}^{\text{DP}}(X, Y)$ and its gradient in efficient vector-matrix notation. Note that the algorithm is closely related to subSDTW from

Chapter 7 and dDTW from Chapter 9. However, for completeness we repeat the algorithm at this place and follow the notation introduced in Section 10.2, such that this chapter can be read stand-alone and serves as a mathematical description of the dDTW toolbox, which is further described in Appendix C.

We base our algorithm on a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ that is computed from the input sequences X and Y as follows:

$$\mathbf{C}(n, m) = c(x_n, y_m) . \quad (10.22)$$

The cells (n, m) of the cost matrix denote the vertices of the graph.

10.4.1 Forward Recursion

As outlined in Section 10.3, we iteratively want to compute accumulated path costs $\mathbf{D}(p)$ for all graph vertices $p = (n, m) \in \mathcal{I}$. We collect these accumulated path costs in an accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$. To this end, for every cell $(n, m) \in \mathcal{I}$ we define a vector of accumulated incoming cost $\mathbf{f}^{(n, m)} \in \mathbb{R}^{S+1}$, with elements $[1 : S]$ used for the cost of incoming steps/edges $\theta_{((n-i_s, m-j_s) \rightarrow (n, m))}$, and element $S+1$ used for cost $w_{\text{start}}^{(n, m)}$ from a start boundary cell. Identical to weighted SDTW [168],

$$\mathbf{f}_s^{(n, m)} := \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I} , \\ \mathbf{D}(n - i_s, m - j_s) + \mathbf{W}(n, m, s) \cdot \mathbf{C}(n, m), & \text{else} \end{cases} \quad (10.23)$$

denotes the cost of incoming steps (i_s, j_s) with $s \in [1 : S]$. The additional contribution in case (n, m) is in the start boundary conditions is given by

$$\mathbf{f}_{S+1}^{(n, m)} := \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}} , \\ w_{\text{start}}^{(n, m)}, & \text{else ,} \end{cases} \quad (10.24)$$

where we usually have $w_{\text{start}}^{(n, m)} = \mathbf{C}(n, m)$, but it can also be used to modify the current path cost at start boundary cells to prevent alignment collapse to degenerate paths in the case of subsequence alignment (see Chapter 7). As in classical SDTW [30], we compute the minimum over cost values from incoming steps and specify the forward recursion, corresponding to (10.19):

$$\mathbf{D}(n, m) = \mu \left(\mathbf{f}^{(n, m)} \right) . \quad (10.25)$$

As discussed in Sections 10.2 and 10.3, the dDTW graph has a highly structured topology that allows for efficient cost computation using DP techniques. In particular, we can define a partial order $\prec$ on the vertices $\mathcal{I}$ (the set of cost matrix cells):

$$(n', m') \prec (n, m) \quad \text{if} \quad n' \leq n, \quad m' \leq m, \quad (n', m') \neq (n, m) . \quad (10.26)$$

Using this partial order, a cell (n, m) can only be updated with (10.25) if the values for all previous cells $(n', m') \prec (n, m)$ have already been computed. This can be achieved, for example, by nested for-loops over n and m (see Algorithm 10.1), or by parallel processing along the anti-diagonals of $\mathbf{C}$ (see Appendix C).

For later use in the backward recursion, we save the gradient of the updated local cost $\mathbf{D}(n, m)$ w.r.t. the components of $\mathbf{f}^{(n, m)}$ in a backtracking tensor $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ with entries

$$\mathbf{B}(n, m) := \nabla_{\mu} \left(\mathbf{f}^{(n, m)} \right) \in \mathbb{R}^{S+1}, \quad (10.27)$$

where $\mathbf{B}(n, m, s)$ for $s \in [1 : S]$ corresponds to the cost contributions from incoming steps, and $\mathbf{B}(n, m, S+1)$ corresponds to contribution from the start boundary condition. After updating all cells $(n, m) \in \mathcal{I}$, we can include an additional cost $w_{\text{end}}^{(n, m)}$ at the end boundary conditions, e.g., to stabilize subsequence alignment (see Chapter 7). We then collect all accumulated cost values at cells given by the end boundary condition in a collection $\mathbf{b} := (b^{(n, m)})_{(n, m) \in \mathcal{B}_{\text{end}}}$ with elements

$$b^{(n, m)} := \mathbf{D}(n, m) + w_{\text{end}}^{(n, m)}. \quad (10.28)$$

The total cost $\text{dDTW}(\mathbf{C}) \in \mathbb{R}$ is given by evaluating the minimum function μ over the cost at the end boundary cells specified by $\mathbf{b}$, corresponding to (10.20):

$$\text{dDTW}(\mathbf{C}) = \mu(\mathbf{b}). \quad (10.29)$$

The complete dDTW forward pass is summarized in Algorithm 10.1, yielding a computational complexity of $O(NM)$. In the dDTW toolbox implementation, we replace the nested for-loops over N and M with an efficient variant that parallelizes over anti-diagonals of the cost matrix and requires only $O(N + M)$ serial computations (see Appendix C).

10.4.2 Backward Recursion

Next, we need to calculate the gradient $\mathbf{H} \in \mathbb{R}^{N \times M}$ of the total cost $\text{dDTW}(\mathbf{C})$ w.r.t. the local cost matrix $\mathbf{C}$, with elements defined as $\mathbf{H}(n, m) := \partial \text{dDTW}(\mathbf{C}) / \partial \mathbf{C}(n, m)$. First, we apply the chain rule to separate the contribution of cost $\mathbf{C}(n, m)$ to the accumulated cost $\mathbf{D}(n, m)$ (denoted by $\mathbf{G}(n, m)$), and the contribution of accumulated cost $\mathbf{D}(n, m)$ to the total cost (denoted by $\mathbf{E}(n, m)$) [30, 94]:

$$\mathbf{H}(n, m) = \underbrace{\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n, m)}}_{:= \mathbf{E}(n, m)} \cdot \underbrace{\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{C}(n, m)}}_{:= \mathbf{G}(n, m)} \quad (10.30)$$

with $\mathbf{E}, \mathbf{G} \in \mathbb{R}^{N \times M}$. The value of $\mathbf{E}(n, m)$ quantifies how much the accumulated cost $\mathbf{D}(n, m)$ affects the total cost. The value of $\mathbf{D}(n, m)$ can contribute to the total cost in two different ways: by influencing following cells in the accumulated cost matrix [30], and by contributing to the total cost in case (n, m) is

Algorithm 10.1 DP forward pass for graph-based dDTW

```

1: Inputs:
2: Cost matrix  $\mathbf{C} \in \mathbb{R}^{N \times M}$  indexed by cells  $\mathcal{I}$ 
3: Boundary conditions  $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$  with weights  $w_{\text{start}}^{(n,m)}, w_{\text{end}}^{(n,m)}$ 
4: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
5: Step weights  $\mathbf{W} \in \mathbb{R}^{N \times M \times S}$ 
6: Differentiable minimum function  $\mu : \mathbb{R}^I \rightarrow \mathbb{R}$ 
7: for  $m = 1, \dots, M$  do
8:   for  $n = 1, \dots, N$  do
9:     for  $s = 1, \dots, S$  do
10:       $\mathbf{f}_s^{(n,m)} := \begin{cases} \infty, & \text{if } (n - i_s, m - j_s) \notin \mathcal{I}, \\ \mathbf{D}(n - i_s, m - j_s) + \mathbf{W}(n, m, s) \cdot \mathbf{C}(n, m), & \text{else} \end{cases}$ 
11:    end for
12:     $\mathbf{f}_{S+1}^{(n,m)} := \begin{cases} \infty, & \text{if } (n, m) \notin \mathcal{B}_{\text{start}}, \\ w_{\text{start}}^{(n,m)}, & \text{else} \end{cases}$ 
13:     $\mathbf{D}(n, m) = \mu(\mathbf{f}^{(n,m)})$ 
14:    Save  $\mathbf{B}(n, m) = \nabla \mu(\mathbf{f}^{(n,m)})$  for backward
15:  end for
16: end for
17: for  $(n, m) \in \mathcal{B}_{\text{end}}$  do
18:    $b^{(n,m)} := \mathbf{D}(n, m) + w_{\text{end}}^{(n,m)}$ 
19: end for
20:  $\mathbf{b} := (b^{(n,m)})_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
21:  $\text{dDTW}(\mathbf{C}) = \mu(\mathbf{b})$ 
22: Output: dDTW cost  $\text{dDTW}(\mathbf{C}) \in \mathbb{R}$ , backtracking tensor  $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ , end boundary costs  $\mathbf{b}$ 

```

contained in the set of end boundary cells. We therefore write

$$\mathbf{E}(n, m) = \sum_{s=1}^S \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} \cdot \frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} + [\nabla \mu(\mathbf{b})]^{(n,m)}, \quad (10.31)$$

where the first term in (10.31) denotes influence of a following cell on the total cost and, as in standard SDTW [30], is given by previously computed entries of the $\mathbf{E}$ -matrix:

$$\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n + i_s, m + j_s)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{E}(n + i_s, m + j_s), & \text{else.} \end{cases} \quad (10.32)$$

The second term in (10.31), also found in standard SDTW [30], denotes the influence of the current cell on a following cell and is given by the previously computed backtracking matrix:

$$\frac{\partial \mathbf{D}(n + i_s, m + j_s)}{\partial \mathbf{D}(n, m)} = \begin{cases} 0, & \text{if } (n + i_s, m + j_s) \notin \mathcal{I}, \\ \mathbf{B}(n + i_s, m + j_s, s), & \text{else.} \end{cases} \quad (10.33)$$

Algorithm 10.2 DP backward pass for graph-based dDTW

```

1: Inputs:
2: Backtracking tensor  $\mathbf{B} \in \mathbb{R}^{N \times M \times (S+1)}$ 
3: Start and end boundary conditions  $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$ 
4: End boundary costs  $\mathbf{b} := (b^{(n,m)})_{(n,m) \in \mathcal{B}_{\text{end}}}$ 
5: Step weights  $\mathbf{W} \in \mathbb{R}^{N \times M \times S}$ 
6: Start boundary weights  $w_{\text{start}}^{(n,m)}$ 
7: Step sizes  $\mathcal{S} = \{(i_s, j_s) | s \in [1 : S]\}$ 
8: for  $m = M, \dots, 1$  do
9:   for  $n = N, \dots, 1$  do
10:     $\frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{E}(n+i_s, m+j_s), & \text{else.} \end{cases}$ 
11:     $\frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n,m)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{B}(n+i_s, m+j_s, s), & \text{else.} \end{cases}$ 
12:     $[\nabla \mu(\mathbf{b})]^{(n,m)} := \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n,m)}, & \text{if } (n,m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases}$ 
13:     $\mathbf{E}(n,m) = \sum_{s=1}^S \frac{\partial \text{dDTW}(\mathbf{C})}{\partial \mathbf{D}(n+i_s, m+j_s)} \cdot \frac{\partial \mathbf{D}(n+i_s, m+j_s)}{\partial \mathbf{D}(n,m)} + [\nabla \mu(\mathbf{b})]^{(n,m)}$ 
14:     $\frac{\partial \mathbf{D}(n,m)}{\partial \mathbf{f}_s^{(n,m)}} = \mathbf{B}(n,m,s)$ 
15:     $\frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n,m)} = \begin{cases} \mathbf{W}(n,m,s), & \text{if } s \in [1 : S], \\ \partial w_{\text{start}}^{(n,m)} / \partial \mathbf{C}(n,m), & \text{if } s=S+1 \text{ and } (n,m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases}$ 
16:     $\mathbf{G}(n,m) = \sum_{s=1}^{S+1} \frac{\partial \mathbf{D}(n,m)}{\partial \mathbf{f}_s^{(n,m)}} \cdot \frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n,m)}$ 
17:     $\mathbf{H}(n,m) = \mathbf{E}(n,m) \cdot \mathbf{G}(n,m)$ 
18:   end for
19: end for
20: Output: Gradient  $\mathbf{H} \in \mathbb{R}^{N \times M}$ 

```

The third term in (10.31) denotes potential contribution coming from paths ending in (n, m) , in case (n, m) is in the set of end boundary conditions:

$$[\nabla \mu(\mathbf{b})]^{(n,m)} := \begin{cases} \partial \mu(\mathbf{b}) / \partial b^{(n,m)}, & \text{if } (n,m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else.} \end{cases} \quad (10.34)$$

The contribution of the local cost $\mathbf{C}(n, m)$ on the accumulated cost $\mathbf{D}(n, m)$ is given by the chain rule as follows:

$$\mathbf{G}(n,m) = \sum_{s=1}^{S+1} \frac{\partial \mathbf{D}(n,m)}{\partial \mathbf{f}_s^{(n,m)}} \cdot \frac{\partial \mathbf{f}_s^{(n,m)}}{\partial \mathbf{C}(n,m)}, \quad (10.35)$$

10.5. Examples for Alignment Problems Generalized by dDTW

Param.	DTW	SDTW	sm.DTW	2-1 DTW	subDTW	PM	CTC
$\mathcal{S}$	$\left\{ \begin{pmatrix} (1,0) \\ (0,1) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (1,0) \\ (0,1) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (1,0) \\ (0,1) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (2,1) \\ (1,2) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (1,0) \\ (0,1) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (1,0) \\ (0,1) \\ (1,1) \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} (1,0) \\ (1,1) \\ (1,2) \end{pmatrix} \right\}$
$\mathbf{W}(n, m)$	$[1, 1, 1]^\top$	$[1, 1, 1]^\top$	$[1, 1, 1]^\top$	$[1, 1, 1]^\top$	$[1, 1, 1]^\top$	$[0, 0, 1]^\top$	$\left\{ \begin{bmatrix} [1, 1, 1]^\top \\ [1, 1, \infty]^\top \end{bmatrix} \right\}$
$\mathcal{B}_{\text{start}}$	$\{(1, 1)\}$	$\{(1, 1)\}$	$\{(1, 1)\}$	$\{(1, 1)\}$	$\left\{ \begin{pmatrix} (1,1) \\ \vdots \\ (N,1) \end{pmatrix} \right\}$	$\mathcal{I}$	$\left\{ \begin{pmatrix} (1,1) \\ (1,2) \end{pmatrix} \right\}$
$\mathcal{B}_{\text{end}}$	$\{(N, M)\}$	$\{(N, M)\}$	$\{(N, M)\}$	$\{(N, M)\}$	$\left\{ \begin{pmatrix} (1,M) \\ \vdots \\ (N,M) \end{pmatrix} \right\}$	$\mathcal{I}$	$\left\{ \begin{pmatrix} (N, M-1) \\ (N, M) \end{pmatrix} \right\}$
μ	min	μ_{soft}^γ	$\mu_{\text{smooth}}^\gamma$	min	min	min	μ_{soft}^γ

Table 10.1: Overview of dDTW parameterizations leading to different DTW variants.

where the first term in (10.35) is given by the backtracking matrix

$$\frac{\partial \mathbf{D}(n, m)}{\partial \mathbf{f}_s^{(n, m)}} = \mathbf{B}(n, m, s), \quad (10.36)$$

and the second term in (10.35) is given by either the step weight or, in the case (n, m) is in the start boundary conditions, the derivative of the respective boundary weight w.r.t. the local cost:

$$\frac{\partial \mathbf{f}_s^{(n, m)}}{\partial \mathbf{C}(n, m)} = \begin{cases} \mathbf{W}(n, m, s), & \text{if } s \in [1 : S], \\ \partial w_{\text{start}}^{(n, m)} / \partial \mathbf{C}(n, m), & \text{if } s=S+1 \text{ and } (n, m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else.} \end{cases} \quad (10.37)$$

Note that for standard SDTW we have $w_{\text{start}}^{(n, m)} = \mathbf{C}(n, m)$ and thus $\partial w_{\text{start}}^{(n, m)} / \partial \mathbf{C}(n, m) = 1$ in (10.37). The complete dDTW backward pass is summarized in Algorithm 10.2. As in the forward pass, in the dDTW toolbox implementation we replace the nested for loops and parallelize the computation over anti-diagonals of the cost matrix (see Appendix C).

10.5 Examples for Alignment Problems Generalized by dDTW

In this section, we outline how different alignment problems can be modeled as special cases of the dDTW shortest-path problem. An overview is provided in Table 10.1.

10.5.1 Classical Dynamic Time Warping

The classical DTW problem [98] is defined as a finding the shortest path from the start vertex $\mathcal{B}_{\text{start}} = \{(1, 1)\}$ to the end vertex $\mathcal{B}_{\text{end}} = \{(N, M)\}$, taking only unit steps $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$.

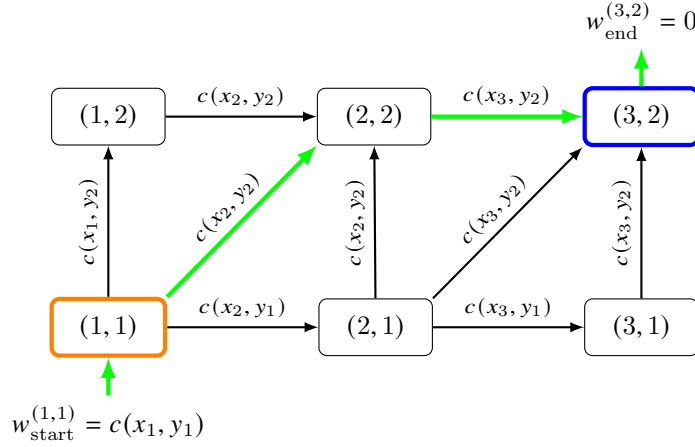


Figure 10.2: Example for a DTW graph with $X = (x_1, x_2, x_3), Y = (y_1, y_2)$. Start vertices are drawn in orange, end vertices are drawn in blue. A valid path is illustrated in green.

Step weights are uniform $\mathbf{W} = \mathbb{1}^{N \times M \times 3}$, the start weight is given as $w_{\text{start}}^{(1,1)} = c(x_1, y_1)$, and there is no end weight $w_{\text{end}}^{(N,M)} = 0$. The minimum operator $\mu = \min$ is the hardmin function. An example for the graph of DTW is illustrated in Figure 10.2.

10.5.2 Differentiable Variants of Classical DTW

In the case of classical DTW, where μ is the hard minimum function, the computation of the shortest-path cost is non-differentiable. To enable differentiability of the problem, several variants of DTW have been proposed, such as soft DTW [30], smooth DTW [61], and sparse DTW [94], all using differentiable approximations of the minimum function. In particular, as described in Section B.2, for a vector $v \in \mathbb{R}^D$ we obtain soft DTW [30] with

$$\mu(v) = \mu_{\text{soft}}^\gamma(v) = \min_{q \in \Delta^D} \langle v, q \rangle + \gamma \langle q, \log q \rangle, \quad (10.38)$$

sparse DTW [94] with

$$\mu(v) = \mu_{\text{sparse}}^\gamma(v) = \min_{q \in \Delta^D} \langle v, q \rangle + \frac{\gamma}{2} \langle q, q - 1 \rangle, \quad (10.39)$$

and smooth DTW [61] with

$$\mu(v) = \mu_{\text{smooth}}^\gamma(v) = \langle v, \nabla \mu_{\text{soft}}^\gamma(v) \rangle, \quad (10.40)$$

where γ is a regularization hyperparameter and Δ^D is the D -dimensional probability simplex. The differentiable variants of DTW can be described by the same graph as classical DTW (see Figure 10.2), as only the minimum function changes. However, note that the DP recursions only solve the global problem if μ satisfies distributivity and associativity, which is the case for the hard- and softmin functions.

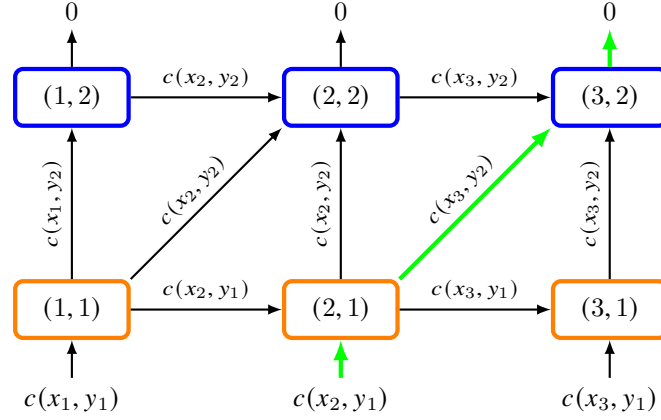


Figure 10.3: Example for a subDTW graph with $X = (x_1, x_2, x_3)$, $Y = (y_1, y_2)$. Start vertices are drawn in orange, end vertices are drawn in blue. A valid path is illustrated in green.

10.5.3 DTW Variants with Different Step Sizes and Step Weights

There exist several variants of DTW and SDTW using different step sizes and step weights [98]. For example, the so-called 2-1 DTW parameterization with step sizes $\mathcal{S} = \{(2, 1), (1, 2), (1, 1)\}$ is widely used in audio synchronization [44, 103] and audio matching [5, 160] to limit excessive warping. Adjusting the step weights $\mathbf{W}$ in the case of very short or long sequences can be used to avoid collapse to trivial cost-minimizing solutions [160, 168].

10.5.4 Subsequence (Soft) DTW

Some applications require finding the occurrence of a short query sequence $Y = (y_1, \dots, y_M)$ in a longer document sequence $X = (x_1, \dots, x_N)$ with $N \gg M$ [5]. In subDTW [98], the boundary conditions are modified such that

$$\mathcal{B}_{\text{start}} = \{(n, 1) \mid n \in [1:N]\} \quad (10.41)$$

$$\mathcal{B}_{\text{end}} = \{(n, M) \mid n \in [1:N]\}. \quad (10.42)$$

Similarly, differentiable variants of subDTW can be obtained using an appropriate choice of μ . An exemplary graph of subDTW and its variants is illustrated in Figure 10.3. We refer to Chapter 7 for a more general formulation of differentiable subsequence alignment, termed subSDTW.

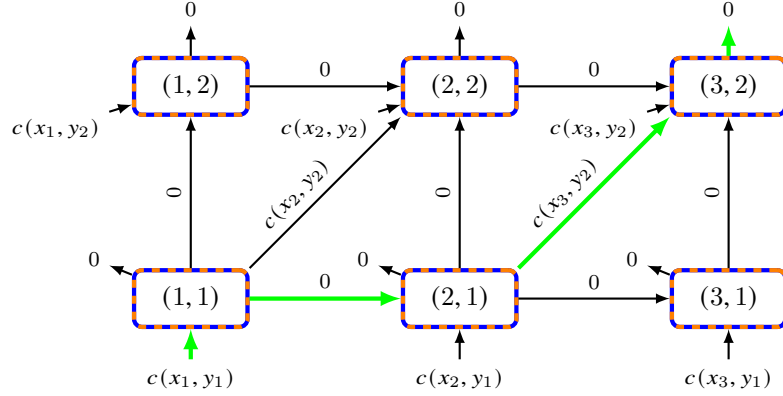


Figure 10.4: Example for a partial matching graph with $X = (x_1, x_2, x_3)$, $Y = (y_1, y_2)$. All vertices $\mathcal{I}$ are start and end vertices. A valid path is illustrated in green.

10.5.5 Partial Matching

Partial matching is a strategy for finding a cost-minimizing alignment that is less constrained than DTW and its variants. The only requirement is a strictly monotonic alignment without imposing step size constraints [2, 45, 98, 99, 118]. Partial matching can be parameterized in the dDTW graph framework by choosing step sizes $\mathcal{S} = \{(1, 0), (0, 1), (1, 1)\}$ and step weights $\mathbf{W}(n, m) = [0, 0, 1]^\top$ for all $(n, m) \in \mathcal{I}$, such that only diagonal steps (strictly monotonic steps) contribute to the alignment cost. Horizontal or vertical steps can be taken with zero cost. As illustrated in the graph of Figure 10.4, a partial match can start and end in any cell, thus $\mathcal{B}_{\text{start}} = \mathcal{B}_{\text{end}} = \mathcal{I}$. Note that partial matching is usually applied to a cost matrix with positive and negative values (as opposed to DTW and SDTW, which usually operate on strictly positive cost matrices), such that the optimum path does not degenerate to a single cost-minimizing cell.

10.5.6 Connectionist Temporal Classification

To represent the CTC loss [59] as a dDTW graph shortest-path problem, we first need to represent the CTC label sequence $Y = (y_1, \dots, y_M)$ as the compressed label sequence $Y^c = (y_1^c, \dots, y_{M^c}^c)$ interleaved with the blank symbol ϵ such that $Y = (\epsilon, y_1^c, \epsilon, \dots, \epsilon, y_{M^c}^c, \epsilon)$ and $M = 2M^c + 1$ (see Section 9.2). The cost function c is the negative log-probability of the labels y_m given the predictions x_n

$$c(x_n, y_m) = -\log p(y_m | x_n). \quad (10.43)$$

Alignment paths are allowed to start either in the blank label or the first actual target symbol, i.e., $\mathcal{B}_{\text{start}} = \{(1, 1), (1, 2)\}$ and end in the blank label or the last target symbol, i.e., $\mathcal{B}_{\text{end}} = \{(N, M-1), (N, M)\}$ with weights $w_{\text{start}}^{(n, m)} = c(x_n, y_m)$ for all $(n, m) \in \mathcal{B}_{\text{start}}$ and $w_{\text{end}}^{(n, m)} = 0$ for all $(n, m) \in \mathcal{B}_{\text{end}}$. The CTC step sizes are strictly monotonic in n and are restricted to the set $\mathcal{S} = \{(1, 0), (1, 1), (1, 2)\}$. The $(1, 2)$

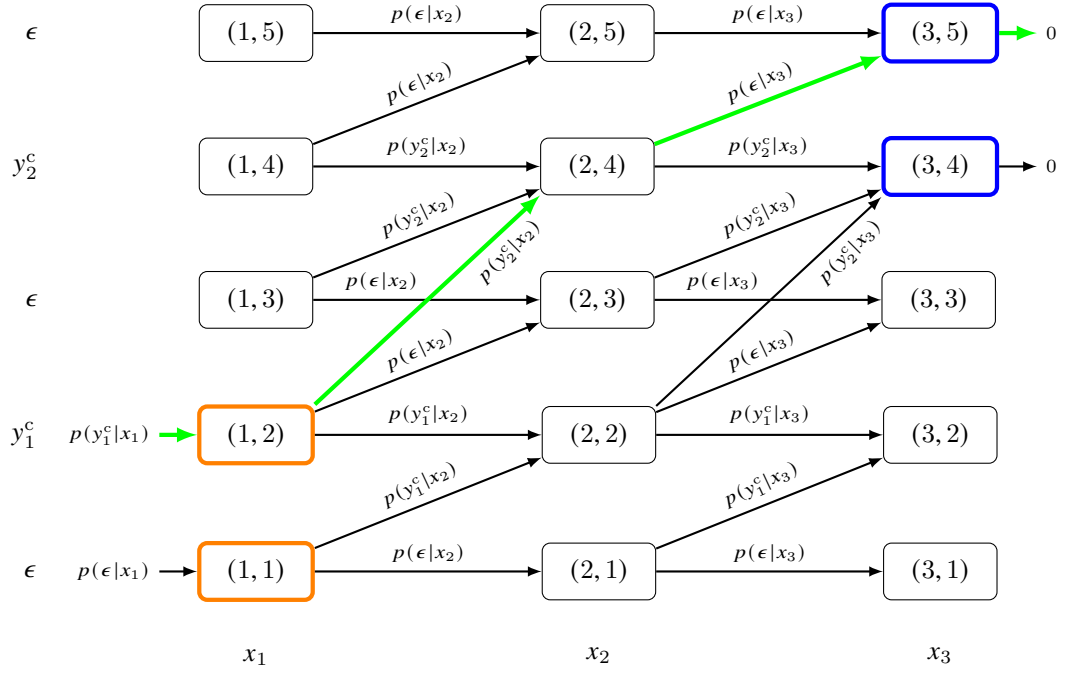


Figure 10.5: Example for a dDTW graph to replicate CTC with $X = (x_1, x_2, x_3)$, $Y = (\epsilon, y_1^c, \epsilon, y_2^c, \epsilon)$. Start vertices are drawn in orange, end vertices are drawn in blue. A valid path is illustrated in green.

step (skipping a target symbol) is only allowed when the skipped symbol is the blank symbol and when the adjacent symbols differ. This is realized through a step weight tensor $\mathbf{W} = \{1, \infty\}^{N \times M \times 3}$, where all allowed steps have a weight of one, and “forbidden” (1, 2)-steps have a weight of infinity. An exemplary graph for CTC is illustrated in Figure 10.5, assuming $y_1^c \neq y_2^c$. More details about the representation of the CTC loss within the dDTW framework are given in Chapter 9.

10.6 Further Notes

In this chapter, we presented a unified graph-based perspective on differentiable sequence alignment, providing a common mathematical framework for the alignment methods discussed throughout this thesis. By formulating sequence alignment as a minimum-cost path problem on a weighted DAG, we showed how different alignment paradigms can be characterized through their step sizes, step weights, boundary conditions, and choice of minimum operator.

A central result of this chapter is the derivation of the DP recursions directly from the global shortest-path formulation. We showed that the global alignment cost can be computed efficiently using DP, provided that the minimum operator satisfies the required associativity and distributivity properties. Building on this result, we derived the corresponding forward and backward recursions for computing the alignment cost and its gradient, forming the mathematical basis of the proposed dDTW toolbox (see Section C).

Chapter 10. A Graph Perspective on Differentiable Alignment

Finally, we demonstrated the generality of the framework by expressing classical DTW, differentiable DTW variants, subsequence alignment, partial matching, and CTC as specific graph parameterizations. Beyond unifying existing methods, the proposed graph perspective provides a systematic way to understand their differences, identify the conditions under which efficient computation is possible, and develop new differentiable alignment methods. In this way, the chapter consolidates the methodological contributions of this thesis and establishes a foundation for future extensions of the dDTW framework.

11 Summary and Future Work

This thesis studied and advanced differentiable alignment techniques in the context of music processing.

In the first part, we grounded alignment in a traditional music synchronization scenario and established a connection to differentiable methods by incorporating learned intermediate representations extracted from transcription networks into the classical DTW-based alignment framework. Through a systematic analysis of features, alignment approaches, and model fine-tuning strategies, we demonstrated that learned representations can substantially improve the temporal accuracy for music synchronization. The practical implications of this integration were substantiated through the curation of the BPSD, a highly consistent multi-version classical music dataset constructed via accurate and robust synchronization based on features from differentiable models.

While in the first part of this thesis, alignment was used as a standalone component, in the second part, it became part of the learning objective. By adopting a differentiable relaxation of DTW, termed SDTW, we incorporated sequence alignment into gradient-based optimization and employed it as a training criterion for DNNs with weakly aligned data. Beyond applying SDTW, this thesis systematically analyzed its optimization behavior, mathematical structure, and characteristic artifacts. Classical alignment concepts, such as weighted steps and subsequence matching, were translated into the differentiable regime, thereby extending the expressive capacity of differentiable alignment losses.

Moving to a more abstract level, this thesis then examined differentiable alignment as a general computational principle. By establishing the theoretical relationship between SDTW and the widely used CTC loss, we revealed that these approaches, often treated as conceptually distinct, can be understood within a shared formal framework. The introduction of dDTW provided such a generalized formulation, subsuming both SDTW and CTC as special cases, thereby making their structural commonalities and fundamental differences explicit. Finally, by adopting a graph-theoretic perspective on differentiable sequence alignment, we formalized validity conditions for the underlying DP recursions used by all alignment methods throughout this thesis. This perspective not only clarifies the algorithmic foundations of existing approaches, but also establishes a systematic framework in which new differentiable alignment objectives can be derived and analyzed.

We conclude with an outlook and discussion of future research directions, which can be grouped into three complementary areas: applications, extensions of differentiable alignment methods, and broader conceptually different directions.

With respect to **applications**, promising avenues include the use of differentiable alignment techniques in cross-version scenarios, where multiple performances or representations of the same piece must be synchronized reliably to enable learning from the given data. Further improvements in synchronization accuracy can be expected from a deeper investigation of onset and offset modeling, particularly in the context of music transcription, where accurate onset representations are critical for alignment quality and offsets are often ill-posed. In addition, multi-task MIR applications offer opportunities to jointly learn alignment-invariant representations that support several downstream tasks within a unified framework.

A second group of directions concerns **extensions of SDTW** itself. One important aspect is the systematic investigation of alternative minimum operators beyond softmin, e.g., sparsemin, and their practical impact on alignment quality and training stability. In particular, an interesting direction is the comparison of convexly regularized minimum functions such as softmin or sparsemin (as lower bounds to the hard minimum) and the smoothmin function (as an upper bound). Another key challenge lies in the development of highly optimized CPU and GPU implementations, comparable in efficiency to established solutions for losses such as CTC, in order to facilitate large-scale applications.

Finally, several **broadier research directions** emerge from this thesis. A limitation of SDTW-based synchronization arises when multiple events occur within a single feature frame. In symbolic representations, a target vector may contain several simultaneously active notes, for example the pitches of a chord. Since SDTW aligns entire feature vectors, all events contained in such a vector are matched to a single prediction frame. In practice, however, chords are often played with slight temporal offsets (e.g., due to arpeggio or playing inaccuracies). Because SDTW cannot resolve these timing differences within a frame, the aligned sequences do not match exactly. In classical DTW, several methods were proposed to perform a local “snapping”, i.e., to align target events to the most suitable predictions within a temporal neighborhood. Transferring these approaches to the differentiable domain can further increase the training accuracy with SDTW. Another promising direction is the investigation of optimal transport as an alternative alignment paradigm. In contrast to the order-preserving nature of SDTW and CTC, optimal transport enforces mass preservation and may offer complementary advantages for certain alignment problems.

Appendix

A Convex Optimization

In this appendix, we describe a method to derive differentiable approximations of functions based on the convex conjugate. We first define some general properties of functions in Section A.1 before describing the convex conjugate in Section A.2. As examples, we provide a full derivation of the softmin and sparsemin functions in Appendix B. The following section closely follows [16, 21, 94].

A.1 Prerequisites and Definitions

Let $f : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\pm\infty\}$ be a function with domain $\text{dom}(f) := \{x \mid f(x) < \infty\}$. To eliminate certain degenerate cases in optimization, we require functions to be proper.

Definition A.1: Properness [123, p. 5]

A function $f : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\pm\infty\}$ is called proper if the following conditions are fulfilled:

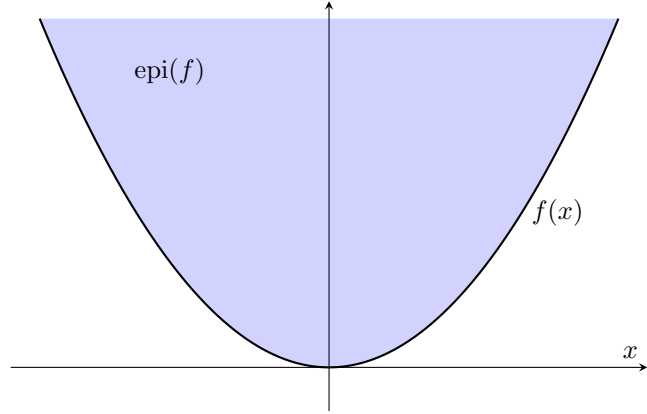
1. $f(x) > -\infty$ for all $x \in \mathbb{R}^D$,
2. $\exists x_0 \in \mathbb{R}^D$ such that $f(x_0) < +\infty$.

Intuitively, the first condition prevents the function from attaining the value minus infinity, which would render minimization problems ill-posed. The second condition ensures that the function is not identically equal to $+\infty$, i.e., that there exists at least one point with a finite function value. Together, these conditions guarantee that the optimization problem is nontrivial and admits at least one feasible point.

Studying the graph of functions provides a geometric intuition to the properties of a function. In convex analysis, this intuition is often formalized by considering sets associated with a function rather than the function itself. To this end, we define the epigraph, following [21].

Appendix

Figure A.1: Illustration of a function f and its epigraph $\text{epi}(f)$.



Definition A.2: Epigraph [21, p. 75]

The epigraph of a function $f : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\pm\infty\}$ is defined as:

$$\text{epi}(f) = \{(x, r) \in \mathbb{R}^D \times \mathbb{R} \cup \{\infty\} \mid r \geq f(x)\}.$$

A function f is called *closed* if and only if its epigraph is a closed set.

In other words, the epigraph of a function is the region in $\mathbb{R}^D \times \mathbb{R}$ that lies on or above the graph of f . We call the epigraph a *closed set* if it contains all its limit points. Many important properties of functions, such as convexity and closedness, can be equivalently characterized by corresponding geometric properties of their epigraphs. An illustration of the epigraph is provided in Figure A.1.

A central property of functions when considering convex optimization is, of course, convexity. Convexity imposes a global structure on the function by requiring that the function values along any line segment between two points lie below the linear interpolation of the endpoint values.

Definition A.3: Convex Function [21, p. 7, Def. 3.1.1]

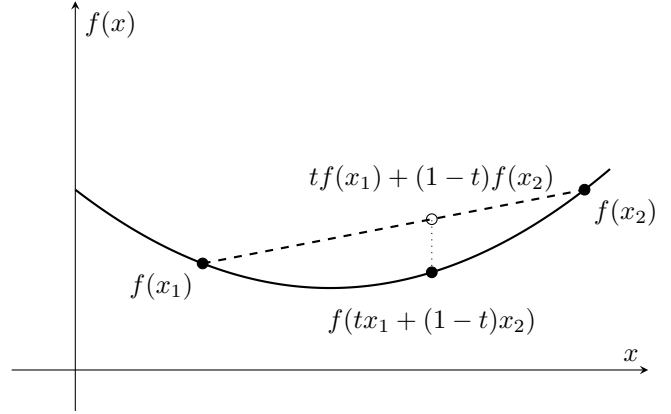
A function $f : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\infty\}$ is called convex if $\text{dom}(f)$ is a convex set and if for all $0 \leq t \leq 1$ and for all $x_1, x_2 \in \text{dom}(f)$:

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Geometrically, convexity means that the graph of the function lies below every line connecting two points on the graph. This property rules out the presence of local minima that are not global and is therefore fundamental for the tractability of optimization problems. An illustration of the convexity property is provided in Figure A.2.

Extending the general definition of convexity, we now explicitly define strong convexity [16]. While convexity only requires that a function lies below all lines connecting two points on the function, strong

Figure A.2: Illustration of the convexity property of a function f .



convexity strengthens this condition by enforcing a quadratic lower bound on the function. In this sense, strong convexity formalizes the idea that the function possesses a strictly positive and uniform amount of curvature everywhere, preventing flat regions and ensuring that the function bends upward at least at a fixed rate. Since the quadratic term in Definition A.4 is nonnegative for all $\beta > 0$, the inequality implies the general convexity condition from before, and thus every β -strongly convex function is also convex.

Definition A.4: β -Strong-Convexity [16, p. 410, Def. 15.7]

A differentiable function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is β -strongly convex for $\beta > 0$ if and only if for all $u, v \in \mathbb{R}^D$:

$$f(u) \geq f(v) + \langle \nabla f(v), u - v \rangle + \frac{\beta}{2} \|u - v\|_2^2.$$

In other words, the function f lies above its linear approximation by an amount specified by β . Intuitively, strong convexity prevents the function from being too flat. Even locally, the function must curve upwards at least quadratically. As a consequence, strongly convex functions admit a unique minimizer.

The continuous differentiability of a function is implied by the smoothness property [16], which controls how rapidly the gradient of the function can change.

Definition A.5: β -Smoothness [16, p. 406, Def. 15.3]

A differentiable function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is β -smooth for $\beta > 0$ if for all $u, v \in \mathbb{R}^D$:

$$\|\nabla f(u) - \nabla f(v)\|_2 \leq \beta \|u - v\|_2.$$

In other words, the variation of the gradient ∇f has a global linear bound. Geometrically, this means that the function cannot bend arbitrarily sharply. Smoothness thus provides an upper bound on the curvature of the function. Note that β -smoothness implies continuous differentiability, as for $u \rightarrow v$ we have

$$\|\nabla f(u) - \nabla f(v)\|_2 \leq \beta \|u - v\|_2 \rightarrow 0 \quad (\text{A.1})$$

Appendix

and thus ∇f is continuous.

Having described basic properties of functions, we now define the indicator function and support function as two special functions that will be used in the calculation of the convex conjugate.

Definition A.6: Indicator Function [21, p. 68]

For a set $C \subseteq \mathbb{R}^D$, the indicator function $I_C : \mathbb{R}^D \rightarrow \{0, \infty\}$ is defined as:

$$I_C(x) := \begin{cases} 0, & x \in C \\ \infty, & x \notin C. \end{cases}$$

In short, the indicator function is zero if the argument is an element of the given set, and infinity if the argument is not in the set. This construction allows constraints of the form $x \in C$ to be incorporated directly into optimization problems by adding the indicator function to an objective, thereby transforming constrained problems into unconstrained ones, e.g.,

$$\min_{x \in C} f(x) = \min_{x \in \mathbb{R}^D} (f(x) + I_C(x)). \quad (\text{A.2})$$

Definition A.7: Support Function [21, p. 81]

For a set $C \subseteq \mathbb{R}^D$, the support function $S_C : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\infty\}$ is defined as:

$$S_C(x) := \sup_{q \in C} \langle x, q \rangle.$$

Intuitively spoken, the support function measures the extent of the set C in the direction of x , i.e., it captures how far the set stretches along that direction.

A.2 The Convex Conjugate and its Properties

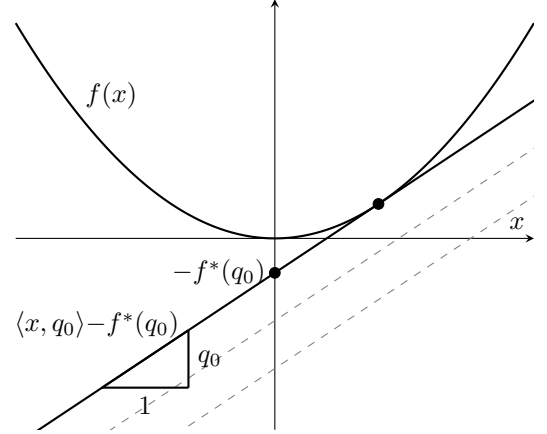
Having defined basic properties of a function f , we now define its convex conjugate f^* . The convex conjugate transform is of central importance in convex optimization [21] and allows for constructing dual problems in optimization theory. The convex conjugation re-expresses a function as the supremum of affine functions and characterizes how well these functions approximate it from below (see Figure A.3) [21].

Definition A.8: Convex Conjugate (Legendre–Fenchel transform) [21, p. 91]

The convex conjugate $f^* : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\infty\}$ of a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is defined by:

$$f^*(q) := \sup_{x \in \text{dom}(f)} (\langle x, q \rangle - f(x)).$$

Figure A.3: Illustration of the convex conjugate. For a slope q , the convex conjugate is the negative value of the y-axis intersection of the tightest line with slope q touching the function f from below. Here, we illustrate lines with a fixed slope q_0 .



Intuitively spoken, the convex conjugate f^* of a function f quantifies how the function can be approximated by affine functions with slopes q . Geometrically, it is defined via the negative y-axis intercepts of the tangent (supporting) lines with slope q that touch the function f from below (see Figure A.3 for an illustration). Among all affine functions with fixed slope q , the supporting line is the highest one that does not cross the graph of f , and its intercept uniquely determines the value of $f^*(q)$.

The convex conjugate f^* has several important properties that make it a central tool in convex analysis and optimization. First of all, the definition ensures that f^* is always at least weakly convex [21].

Theorem A.9: Convexity of the Convex Conjugate [21, p. 91]

f^* is a convex function, even if f is nonconvex.

Applying the convex conjugate transform twice yields a function f^{**} , also called the biconjugate, defined on the primal space [21]. The biconjugate can be interpreted as the pointwise supremum of all affine underestimators of the original function, satisfying convexity and closedness (meaning the epigraph is a closed set) [21]. Thus, equality of the biconjugate f^{**} and the original function f thus only in a special case:

Theorem A.10: Equality of a Function and its Biconjugate (Fenchel–Moreau Theorem) [21, p. 94]

For any real-valued function $f : \mathbb{R}^D \rightarrow \mathbb{R} \cup \{\infty\}$ it holds that $f = f^{**}$ if and only if f is proper, closed (i.e., the epigraph $\text{epi}(f)$ is a closed set), and convex.

This result, known as the Fenchel–Moreau Theorem, provides a precise condition under which the convex conjugate transform is involutive, i.e., a self-inverse function. One example of a function that is equal to its biconjugate is the maximum function, as it satisfies the necessary conditions of properness, a closed epigraph, and (weak) convexity.

Regarding differentiability, the convex conjugate also preserves certain properties in a dual sense. In particular, strong convexity of a function translates into smoothness of its conjugate [16].

Theorem A.11: Smoothness of the Convex Conjugate [16, p. 347, Prop. 13.4]

If f is $\frac{1}{\beta}$ -strongly convex over $\text{dom}(f)$, then f^* is β -smooth over $\text{dom}(f^*)$, thus f^* has a continuous derivative over $\text{dom}(f^*)$.

This duality between convexity and smoothness is a key principle in ensuring continuous differentiability via convex regularization, as presented in Appendix B.

B Differentiable Minimum Functions

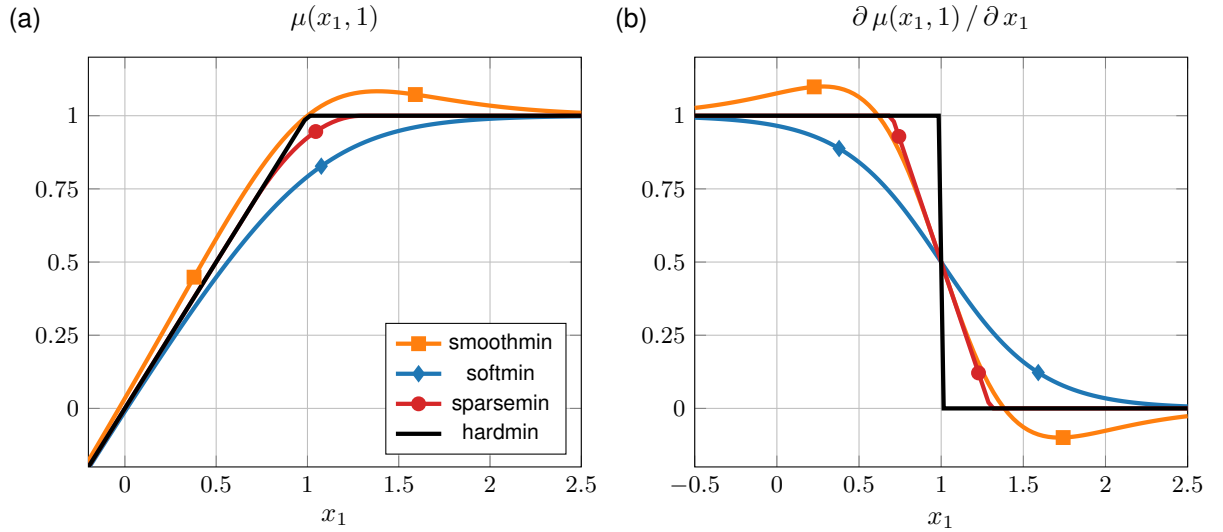


Figure B.1: Overview of (a) differentiable minimum functions μ and (b) their partial derivatives. We use a temperature parameter $\gamma = 0.3$ for all functions.

This section contains derivations for differentiable minimum functions. Building upon results from convex optimization (see Appendix A), we derive in Section B.1 a general formulation for differentiable minimum functions obtained via convex regularization of the hard minimum function. In Sections B.2 and B.3, we use these results and derive explicit formulations for entropic regularization (yielding softmin) [30] and quadratic regularization (yielding sparsemin) [94], respectively. We conclude with a brief review of the smoothmin function [61] as another type of differentiable minimum function beyond convex regularization. An overview of these differentiable minimum functions and their gradients is provided in Figure B.1.

Note that while in Chapter 4 we defined the differentiable minimum functions μ over index sets, in the following we define them over vectors in $\mathbb{R}^D$, e.g., $\mu : \mathbb{R}^D \rightarrow \mathbb{R}$. For brevity, in the case of scalar inputs $a, b \in \mathbb{R}$, we use the shorthand notation $\mu(a, b)$.

B.1 Derivation of a Differentiable Minimum Function via Convex Regularization

In this section, we describe how to derive a differentiable approximation of the maximum function, following the framework of [16]. Although our ultimate goal is to obtain smooth approximations of the minimum function, we begin with the maximum function because it is convex, whereas the minimum function is concave. This convexity property is essential for the construction and interpretation of the convex conjugate, as defined in Appendix A.

Accordingly, we take the maximum function as the primal function f and pass to the dual space by computing its convex conjugate f^* , as defined in Definition A.8. By Theorem A.9, the conjugate f^* is at

Appendix

least weakly convex. To ensure strong convexity in the dual space, we exploit the fact that the sum of a weakly convex and a strongly convex function is always strongly convex. To this end, we add a strongly convex regularizer $\Omega : \mathbb{R}^D \rightarrow \mathbb{R}$ to the conjugate, yielding the strongly convex function $f_\Omega^* := f^* + \Omega$. Applying the convex conjugate once more, we return to the primal space and obtain $f_\Omega := (f_\Omega^*)^*$, as defined in Definition A.8. When using $f = \max$, the resulting function $f_\Omega = \max_\Omega$ is strongly convex and has a continuous gradient (Theorem A.11).

Note that, in a last step, we obtain a differentiable approximation of the minimum function by exploiting the identity between minimum and maximum $\min_\Omega(x) := -\max_\Omega(-x)$ [94].

With $x \in \mathbb{R}^D$, set

$$f(x) = \max(x), \quad (\text{B.1})$$

which is a proper, closed, and convex function with $\text{dom}(f) = \mathbb{R}^D$. Taking the convex conjugate (Definition A.8) yields

$$f^*(q) = \sup_{x \in \text{dom}(f)} (\langle x, q \rangle - f(x)) \quad (\text{B.2})$$

$$= \sup_{x \in \mathbb{R}^D} (\langle x, q \rangle - \max(x)) \quad (\text{B.3})$$

$$= \begin{cases} 0, & q \in \Delta^D \\ \infty, & q \notin \Delta^D \end{cases} = I_{\Delta^D}(q), \quad (\text{B.4})$$

where $\Delta^D := \{x \in [0, 1]^D \mid \sum_{d=1}^D x_d = 1\}$ is the D -dimensional probability simplex. In (B.4), we have used that $\langle x, q \rangle \leq \max(x)$ for $q \in \Delta^D$, where equality is achieved if the probability mass in q is concentrated on the maximal coordinate of x . Since the convex conjugate guarantees convexity (Theorem A.9) but f^* exhibits a plateau for $q \in \Delta^D$, we know that f^* is weakly convex. To ensure strong convexity in the dual space, we next add a strongly convex regularizer Ω (the sum of a weakly and a strongly convex function is always strongly convex):

$$f_\Omega^*(q) := I_{\Delta^D}(q) + \Omega(q), \quad (\text{B.5})$$

defining the Ω -regularized convex conjugate as $f_\Omega^*(q)$. Last, we compute the convex conjugate of $f_\Omega^*(q)$ to go back to the primal space (Definition A.8):

$$(f_\Omega^*)^*(x) = \sup_{q \in \text{dom}(f_\Omega^*)} (\langle x, q \rangle - f_\Omega^*(q)) \quad (\text{B.6})$$

$$= \sup_{q \in \mathbb{R}^D} (\langle x, q \rangle - I_{\Delta^D}(q) - \Omega(q)) \quad (\text{B.7})$$

$$= \sup_{q \in \mathbb{R}^D} \begin{cases} -\infty, & q \notin \Delta^D \\ \langle x, q \rangle - \Omega(q), & q \in \Delta^D \end{cases} \quad (\text{B.8})$$

$$= \max_{q \in \Delta^D} (\langle x, q \rangle - \Omega(q)) \quad (\text{B.9})$$

$$= f_\Omega(x) =: \max_\Omega(x). \quad (\text{B.10})$$

Because $f_\Omega^*(q)$ is strongly convex, $f_\Omega(x) = \max_\Omega(x)$ is smooth, i.e., it has a continuous derivative (Theorem A.11). Following [94], an expression for differentiable minimum functions $\min_\Omega : \mathbb{R}^D \rightarrow \mathbb{R}$ is directly obtained as:

$$\min_\Omega(x) = -\max_\Omega(-x) \quad (\text{B.11})$$

$$= -\max_{q \in \Delta^D} (\langle -x, q \rangle - \Omega(q)) \quad (\text{B.12})$$

$$= \min_{q \in \Delta^D} (\langle x, q \rangle + \Omega(q)). \quad (\text{B.13})$$

In the next sections, we will give concrete examples for the strongly convex regularizer Ω , namely the negative entropy function (yielding softmin, see Section B.2) and the squared ℓ_2 norm (yielding sparsemin, see Section B.3).

B.2 Entropic Regularization: Softmin

In the following section, we discuss entropic regularization, yielding the widely-used softmin function [30]. We first explicitly derive the softmin formulation resulting from the convex regularization approach, before defining and proving several important properties of the softmin function. The softmin function and its gradient are illustrated for different temperature parameters γ in Figure B.2.

B.2.1 Derivation of the Log-Sum-Exp Formulation

A popular choice for the convex regularizer Ω introduced in Section B.1 is the negative Shannon entropy (“negentropy”) $\Omega_{\text{soft}}(q) = \gamma \langle q, \log q \rangle = \gamma \sum_i q_i \log q_i$, where γ is a parameter controlling the regularization strength. In the following section, we show how to derive the softmin function $\mu_{\text{soft}}^\gamma : \mathbb{R}^D \rightarrow \mathbb{R}$ for negentropy regularization.

Appendix

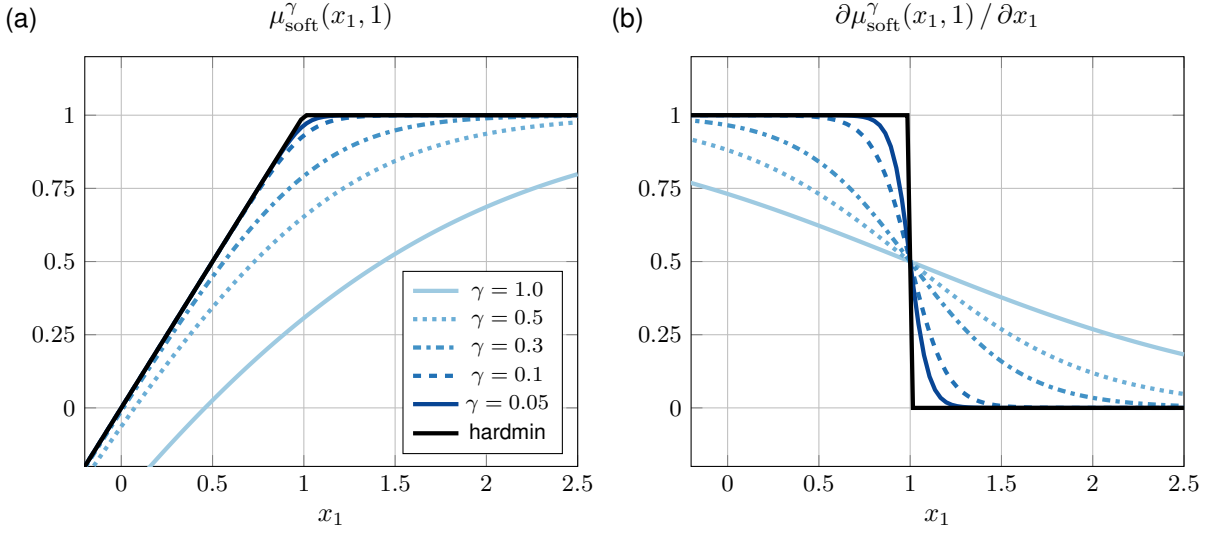


Figure B.2: Overview of (a) the softmax function μ_{soft}^γ and (b) its partial derivative for different temperature parameters γ .

We start off by writing out μ_{soft}^γ in the form with an additive entropic regularizer

$$\mu_{\text{soft}}^\gamma(x) = \min_{q \in \Delta^D} (\langle x, q \rangle + \gamma \langle q, \log q \rangle) \quad (\text{B.14})$$

$$= \min_{q \in \Delta^D} \left(\sum_i q_i x_i + \gamma \sum_i q_i \log q_i \right) \quad (\text{B.15})$$

$$= \min_{q \in \Delta^D} f(q; x), \quad (\text{B.16})$$

where $\Delta^D = \{q \in \mathbb{R}^D \mid q_i \geq 0, \sum_{i=1}^D q_i = 1\}$ denotes the D -dimensional probability simplex and f denotes the objective function for a fixed $x \in \mathbb{R}^D$:

$$f(q; x) = \sum_i q_i x_i + \gamma \sum_i q_i \log q_i. \quad (\text{B.17})$$

Re-writing (B.16) for $q \in \mathbb{R}^D$, we rephrase the optimization problem as:

$$\min_{q \in \mathbb{R}^D} f(q; x) \quad \text{subject to} \quad \begin{cases} h_j(q) = 0 \\ g_i(q) \leq 0 \end{cases} \quad (\text{B.18})$$

with equality constraints $h_j(q) = 0$ and inequality constraints $g_i(q) \leq 0$ [112]. In particular, to satisfy a solution for $q \in \Delta^D$, we require one equality constraint

$$h(q) = \left(\sum_{i=1}^D q_i \right) - 1 = 0, \quad (\text{B.19})$$

and D inequality constraints

$$g_i(q) = -q_i \leq 0, \quad i = 1, \dots, D. \quad (\text{B.20})$$

In the following, we will transform the constrained optimization problem from (B.18) into an unconstrained problem using the method of Lagrange multipliers.

Introducing Lagrange multipliers $\lambda \in \mathbb{R}$ for the equality constraint $h(q)$ and multipliers $\mu_i \geq 0$ for the inequality constraints $g_i(q)$, we obtain the Lagrangian $\mathcal{L}$ as [112]

$$\mathcal{L}(q, \lambda, \mu) = f(q; x) + \lambda h(q) + \sum_{i=1}^D \mu_i g_i(q) \quad (\text{B.21})$$

$$= \sum_{i=1}^D q_i x_i + \gamma \sum_{i=1}^D q_i \log q_i + \lambda h(q) + \sum_{i=1}^D \mu_i g_i(q) \quad (\text{B.22})$$

$$= \sum_{i=1}^D q_i x_i + \gamma \sum_{i=1}^D q_i \log q_i - \lambda \left(1 - \sum_{i=1}^D q_i \right) - \sum_{i=1}^D \mu_i q_i. \quad (\text{B.23})$$

Next, we formulate the Karush–Kuhn–Tucker (KKT) conditions, which characterize optimal solutions of constrained optimization problems [112]. In particular, the KKT conditions require stationarity of the Lagrangian with respect to the primal variables, primal feasibility (the candidate solution satisfies all original equality and inequality constraints), dual feasibility (nonnegativity of the Lagrange multipliers associated with inequality constraints), and complementary slackness (only constraints that are active at the optimum influence the stationarity condition, while inactive constraints have zero multipliers). For the described problem, the KKT conditions are for all $i \in [1 : D]$ [112]:

$$\frac{\partial \mathcal{L}}{\partial q_i} = x_i + \gamma (1 + \log q_i) + \lambda - \mu_i = 0 \quad (\text{stationarity}) \quad (\text{B.24})$$

$$g_i(q) = -q_i \leq 0 \quad (\text{primal feasibility}) \quad (\text{B.25})$$

$$h(q) = \left(\sum_{i=1}^D q_i \right) - 1 = 0 \quad (\text{primal feasibility}) \quad (\text{B.26})$$

$$\mu_i \geq 0 \quad (\text{dual feasibility}) \quad (\text{B.27})$$

$$\mu_i g_i(q) = 0 \quad (\text{complementary slackness}) \quad (\text{B.28})$$

For the discussed softmin problem, the KKT conditions are both necessary and sufficient for optimality, since the problem in (B.14) is convex (guaranteed by the derivation via the convex conjugate, see Theorem A.9), the feasible set is convex (the probability simplex), and there exists at least one point strictly satisfying all equality and inequality constraints (e.g., $q_i = 1/D$ for $i \in [1 : D]$) [112].

In the following, we now solve the optimization problem, fulfilling the KKT conditions. We denote the optimal solutions for q as q^* . Assuming an optimal solution for $q_i^* = 0$, we would have $\lim_{q_i \rightarrow 0} \gamma (1 + \log q_i) = -\infty$ and a stationary point could not be achieved in (B.24) for any finite λ and μ_i .

Appendix

Thus, all solutions have to fulfill $q_i^* > 0$ for $i \in [1 : D]$, complementary slackness from (B.28) implies $\mu_i = 0$ for all $i \in [1 : D]$, and stationarity from (B.24) reduces to

$$x_i + \gamma (1 + \log q_i) + \lambda = 0. \quad (\text{B.29})$$

Solving (B.29) for q_i yields

$$\log q_i = - \left(\frac{x_i + \lambda}{\gamma} + 1 \right) \quad (\text{B.30})$$

$$q_i = e^{\frac{-x_i}{\gamma}} \cdot e^{\frac{-\lambda-\gamma}{\gamma}}, \quad (\text{B.31})$$

which fulfills the previous condition that all $q_i^* > 0$. Inserting (B.31) in the equality constraint (B.26), we obtain

$$\left(\sum_{i=1}^D e^{\frac{-x_i}{\gamma}} \cdot e^{\frac{-\lambda-\gamma}{\gamma}} \right) - 1 = 0, \quad (\text{B.32})$$

and thus

$$e^{\frac{-\lambda-\gamma}{\gamma}} = \frac{1}{\sum_{i=1}^D e^{\frac{-x_i}{\gamma}}}. \quad (\text{B.33})$$

Substituting (B.33) back to (B.31), we obtain the optimal solution

$$q_i^* = \frac{e^{\frac{-x_i}{\gamma}}}{\sum_{i=1}^D e^{\frac{-x_i}{\gamma}}}, \quad (\text{B.34})$$

also known as the softargmin distribution. Inserting (B.34) back into the initial formulation μ_{soft}^γ from (B.14) yields

$$\mu_{\text{soft}}^\gamma(x) = \sum_i q_i^* x_i + \gamma \sum_i q_i^* \log q_i^* \quad (\text{B.35})$$

$$= \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} x_i + \gamma \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} \log \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} \quad (\text{B.36})$$

$$= \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} x_i + \gamma \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} \left(\frac{-x_i}{\gamma} - \log \sum_j e^{-x_j/\gamma} \right) \quad (\text{B.37})$$

$$= \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} x_i - \sum_i \frac{e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} x_i - \gamma \frac{\sum_i e^{-x_i/\gamma}}{\sum_j e^{-x_j/\gamma}} \cdot \log \sum_j e^{-x_j/\gamma} \quad (\text{B.38})$$

$$= -\gamma \log \sum_i e^{-x_i/\gamma}, \quad (\text{B.39})$$

which concludes the derivation of the softmin expression. The optimum solutions q_i^* can be interpreted as the gradient of the softmin function, i.e.,

$$\frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \frac{e^{-x_i/\gamma}}{\sum_{i=1}^D e^{-x_i/\gamma}} = q_i^* . \quad (\text{B.40})$$

B.2.2 Properties of the Softmin Function

In the following section, we describe and prove several important properties of the softmin function, some of which will be necessary for deriving valid DP recursions in Chapters 4 to 10.

Theorem B.1: Properties of the Softmin Function

Let $x = \{x_1, \dots, x_D\} \in \mathbb{R}^D$, $\tilde{x} = x \cup \{\infty\}$, and $a, b, c \in \mathbb{R}$. The softmin function $\mu = \mu_{\text{soft}}^\gamma : \mathbb{R}^D \rightarrow \mathbb{R}$ has the following properties:

- (a) **Boundedness:** $\min(x) \geq \mu(x) \geq \min(x) - \gamma \log D$
- (b) **Insensitivity to Infinity:** $\mu(\tilde{x}) = \mu(x)$
- (c) **Commutativity:** $\mu(a, b) = \mu(b, a)$
- (d) **Distributivity:** $\mu(x + c) = \mu(x) + c$
- (e) **Associativity:** $\mu(a, \mu(b, c)) = \mu(a, b, c)$

Proofs

(a) **Boundedness.** Let $x = \{x_1, \dots, x_D\} \in \mathbb{R}^D$ be a set of real numbers. Defining the set of non-minimum indices of x as $\mathcal{E} := \{j \in [1 : D] \mid x_j > \min(x)\}$ and using $e^a > 0$ for all $a \in \mathbb{R}$, we obtain

$$\sum_{i=1}^D e^{-x_i/\gamma} = e^{-\min(x)/\gamma} + \sum_{j \in \mathcal{E}} e^{-x_j/\gamma} \geq e^{-\min(x)/\gamma} . \quad (\text{B.41})$$

Furthermore, for all $i \in [1 : D]$ and $\gamma > 0$ it holds that:

$$\min(x) \leq x_i \quad (\text{B.42})$$

$$e^{\min(x)} \leq e^{x_i} \quad (\text{B.43})$$

$$e^{-\min(x)/\gamma} \geq e^{-x_i/\gamma} , \quad (\text{B.44})$$

Appendix

and thus

$$\sum_{i=1}^D e^{-x_i/\gamma} = e^{-\min(x)/\gamma} + \sum_{j \in \mathcal{E}} e^{-x_j/\gamma} \leq D \cdot e^{-\min(x)/\gamma}. \quad (\text{B.45})$$

Combining (B.41) and (B.45) yields upper and lower bounds for the sum of exponentials:

$$e^{-\min(x)/\gamma} \leq \sum_{i=1}^D e^{-x_i/\gamma} \leq D e^{-\min(x)/\gamma}. \quad (\text{B.46})$$

Applying the operator $-\gamma \log(\cdot)$ to all terms finally yields upper and lower bounds for the softmin function

$$-\gamma \log \left(e^{-\min(x)/\gamma} \right) \geq -\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \geq -\gamma \log \left(D e^{-\min(x)/\gamma} \right) \quad (\text{B.47})$$

$$\min(x) \geq \mu_{\text{soft}}^{\gamma}(x) \geq \min(x) - \gamma \log D, \quad (\text{B.48})$$

as also provided in [94]. $\square$

(b) Insensitivity to Infinity. Let $x = \{x_1, \dots, x_D\}$ and $\tilde{x} = x \cup \{\infty\}$. Then

$$\mu_{\text{soft}}^{\gamma}(\tilde{x}) = -\gamma \log \left(\left(\sum_{i=1}^D e^{-x_i/\gamma} \right) + \underbrace{e^{-\infty/\gamma}}_{=0} \right) = \mu_{\text{soft}}^{\gamma}(x) \quad (\text{B.49})$$

proves insensitivity of $\mu_{\text{soft}}^{\gamma}$ to appending infinity values to the input variable. $\square$

(c) Commutativity. The commutativity property $\mu_{\text{soft}}^{\gamma}(a, b) = \mu_{\text{soft}}^{\gamma}(b, a)$ for $a, b \in \mathbb{R}$ follows directly from the commutativity property of the sum in (B.39). $\square$

(d) Distributivity. Let $x \in \mathbb{R}^D$ and $c \in \mathbb{R}$.

$$\mu_{\text{soft}}^{\gamma}(x + c) = \min_{q \in \Delta^D} \langle x + c, q \rangle + \gamma \langle q, \log q \rangle \quad (\text{B.50})$$

$$= \min_{q \in \Delta^D} \langle x, q \rangle + \langle \mathbb{1}c, q \rangle + \gamma \langle q, \log q \rangle \quad (\text{B.51})$$

$$= \min_{q \in \Delta^D} \langle x, q \rangle + \gamma \langle q, \log q \rangle + c \quad (\text{B.52})$$

$$= \mu_{\text{soft}}^{\gamma}(x) + c, \quad (\text{B.53})$$

where $\mathbb{1}$ is an all-ones vector of size D , and $\langle \mathbb{1}c, q \rangle = c$ for all $q \in \Delta^D$. The proof holds equivalently for all functions of the form $\min_{\Omega}$. $\square$

(e) Associativity. We first prove associativity for the softmin function $\mu_{\text{soft}}^{\gamma}(a, \mu_{\text{soft}}^{\gamma}(b, c)) = \mu_{\text{soft}}^{\gamma}(a, b, c)$ for $a, b, c \in \mathbb{R}$. We then reference a result from Mensch and Blondel [94] indicating that associativity for functions of the form $\min_{\Omega}$ is in general only fulfilled if Ω is an entropic regularizer. Let $a, b, c \in \mathbb{R}$ be

real numbers. Associativity for μ_{soft}^γ can be proven in closed form:

$$\mu_{\text{soft}}^\gamma \left(a, \mu_{\text{soft}}^\gamma(b, c) \right) = -\gamma \log \left(e^{-a/\gamma} + e^{-\frac{-\gamma \log(e^{-b/\gamma} + e^{-c/\gamma})}{\gamma}} \right) \quad (\text{B.54})$$

$$= -\gamma \log \left(e^{-a/\gamma} + e^{\log(e^{-b/\gamma} + e^{-c/\gamma})} \right) \quad (\text{B.55})$$

$$= -\gamma \log \left(e^{-a/\gamma} + e^{-b/\gamma} + e^{-c/\gamma} \right) \quad (\text{B.56})$$

$$= \mu_{\text{soft}}^\gamma(a, b, c). \quad (\text{B.57})$$

To show that associativity is in general not fulfilled for arbitrary regularizers Ω , Mensch and Blondel [94] first assume that $\Omega : \Delta^D \rightarrow \mathbb{R}$ is a regularization function with additive components of the form $\Omega(q) = \gamma \sum_{i=1}^D \omega(q_i)$. They then show that all solutions to the problem

$$\min_{\Omega}(a, \min_{\Omega}(b, c)) = \min_{\Omega}(a, b, c) \quad (\text{B.58})$$

are of the form

$$\omega(q_i) = q_i \log q_i, \quad (\text{B.59})$$

thus associativity holds if and only if Ω is an entropic regularizer [94]. $\square$

B.2.3 Behavior of the Softmin Function for Limit Cases

For limit values of the temperature parameter γ , the softmin function approximates either the hard minimum function ($\gamma \rightarrow 0$) or the mean function plus a constant ($\gamma \rightarrow \infty$). In the following, we will specify and prove the behavior for these limit cases.

Theorem B.2: Behavior of the Softmin Function for Limit Cases

Let $x = \{x_1, \dots, x_D\} \in \mathbb{R}^D$. The softmin function $\mu_{\text{soft}}^\gamma : \mathbb{R}^D \rightarrow \mathbb{R}$ has the following properties for limit cases of the temperature parameter:

- (a) **Hard-minimum limit:** $\lim_{\gamma \rightarrow 0} \mu_{\text{soft}}^\gamma(x) = \min(x)$
- (b) **Mean-dominated limit:** $\mu_{\text{soft}}^\gamma(x) \xrightarrow{\gamma \rightarrow \infty} -\gamma \log D + \frac{1}{D} \sum_{i=1}^D x_i$

Proofs

(a) **Asymptotic Behavior of the Softmin Function for $\gamma \rightarrow 0$:** To derive the asymptotic behavior of the softmin function for $\gamma \rightarrow 0$, we first define the minimum value $x_{\min} = \min(x)$. Then, we can reformulate

Appendix

the exponential term in the softmax function as

$$\sum_{i=1}^D e^{-x_i/\gamma} = e^{-x_{\min}/\gamma} \sum_{i=1}^D e^{-(x_i - x_{\min})/\gamma}. \quad (\text{B.60})$$

Substituting into the softmax expression yields

$$\mu_{\text{soft}}^\gamma(x) = -\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \quad (\text{B.61})$$

$$= -\gamma \log \left(e^{-x_{\min}/\gamma} \sum_{i=1}^D e^{-(x_i - x_{\min})/\gamma} \right) \quad (\text{B.62})$$

$$= x_{\min} - \gamma \log \sum_{i=1}^D e^{-(x_i - x_{\min})/\gamma}. \quad (\text{B.63})$$

For the exponential term, we have

$$\lim_{\gamma \rightarrow 0} e^{-(x_i - x_{\min})/\gamma} = \begin{cases} 1, & \text{if } x_i = x_{\min}, \\ 0, & \text{else.} \end{cases} \quad (\text{B.64})$$

Thus, the asymptotic behavior of the softmax function for $\gamma \rightarrow 0$ becomes

$$\lim_{\gamma \rightarrow 0} \mu_{\text{soft}}^\gamma(x) = x_{\min} - \gamma \log(1 + 0) \quad (\text{B.65})$$

$$= x_{\min} - \gamma \cdot 0 \quad (\text{B.66})$$

$$= \min(x), \quad (\text{B.67})$$

which concludes the proof. $\square$

(b) Asymptotic Behavior of the Softmin Function for $\gamma \rightarrow \infty$: In contrast, when γ takes large values, the softmax function transitions from a minimum-like operator to a mean-like function. This behavior can be characterized through an asymptotic expansion of the softmax function for $\gamma \rightarrow \infty$. Throughout the following derivation, all expressions involving the symbol $\approx$ are to be interpreted in an asymptotic sense, i.e., as truncated series expansions valid up to the specified order in γ^{-1} . To this end, we use polynomial (Taylor) approximations of the exponential function for $a \in \mathbb{R}$:

$$e^a \approx 1 + a + \frac{a^2}{2} + \dots \quad (\text{B.68})$$

and the logarithm:

$$\log(a) \approx (a - 1) - \frac{(a - 1)^2}{2} + \dots \quad (\text{B.69})$$

$$\log(1 + a) \approx a - \frac{a^2}{2} + \dots, \quad (\text{B.70})$$

where higher-order terms are neglected [163]. Using (B.68), we replace the exponential terms and obtain an asymptotic expansion of the softmin function for $\gamma \rightarrow \infty$:

$$-\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \xrightarrow{\gamma \rightarrow \infty} -\gamma \log \sum_{i=1}^D \left(1 - \frac{x_i}{\gamma} + \frac{x_i^2}{2\gamma^2} \right) \quad (\text{B.71})$$

$$= -\gamma \log \left(D + \sum_{i=1}^D \left(-\frac{x_i}{\gamma} + \frac{x_i^2}{2\gamma^2} \right) \right) \quad (\text{B.72})$$

$$= -\gamma \log \left(D - \frac{1}{\gamma} \sum_{i=1}^D x_i + \frac{1}{2\gamma^2} \sum_{i=1}^D x_i^2 \right), \quad (\text{B.73})$$

where the exponential function is truncated at second order, yielding an approximation accurate up to terms of order $\mathcal{O}(\gamma^{-3})$ inside the logarithm. Defining the first- and second-order moments:

$$m_1 = \frac{1}{D} \sum_{i=1}^D x_i \quad (\text{B.74})$$

$$m_2 = \frac{1}{D} \sum_{i=1}^D x_i^2, \quad (\text{B.75})$$

as well as their combination:

$$v = -\frac{1}{\gamma} m_1 + \frac{1}{2\gamma^2} m_2, \quad (\text{B.76})$$

we can rewrite the argument of the logarithm in a more compact form:

$$-\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \xrightarrow{\gamma \rightarrow \infty} -\gamma \log \left(D - \frac{D}{\gamma} m_1 + \frac{D}{2\gamma^2} m_2 \right) \quad (\text{B.77})$$

$$= -\gamma \log (D(1 + v)) . \quad (\text{B.78})$$

Appendix

Expanding the logarithm in powers of v using (B.70) yields:

$$-\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \xrightarrow{\gamma \rightarrow \infty} -\gamma \log (D(1+v)) \quad (\text{B.79})$$

$$= -\gamma \log D - \gamma \left(v - \frac{v^2}{2} \right) \quad (\text{B.80})$$

$$= -\gamma \log D - \gamma \left(-\frac{m_1}{\gamma} + \frac{m_2}{2\gamma^2} - \frac{m_1^2}{2\gamma^2} + \frac{m_1 m_2}{2\gamma^3} - \frac{m_2^2}{8\gamma^4} \right) \quad (\text{B.81})$$

$$\approx -\gamma \log D + m_1 - \frac{1}{2\gamma}(m_2 - m_1^2) + \dots, \quad (\text{B.82})$$

where terms of order $O(\gamma^{-2})$ and smaller are omitted. Finally, with the mean $m_1 = \frac{1}{D} \sum_{i=1}^D x_i$ and the variance $\text{Var}(x) = m_2 - m_1^2$, we obtain the final result for the asymptotic expansion:

$$-\gamma \log \sum_{i=1}^D e^{-x_i/\gamma} \xrightarrow{\gamma \rightarrow \infty} -\gamma \log D + \frac{1}{D} \sum_{i=1}^D x_i - \frac{1}{2\gamma} \text{Var}(x), \quad (\text{B.83})$$

Thus, for $\gamma \rightarrow \infty$, the softmax function becomes a mean-like function with an entropic bias of order $O(\gamma)$ and a variance-dependent correction of order $O(\gamma^{-1})$. However, note that the leading term $-\gamma \log D$ is independent of x and therefore does not affect the gradient. $\square$

B.2.4 Behavior of the Softmin Gradient Function for Limit Cases

Next, we study the gradient of the softmax function in the limit cases $\gamma \rightarrow 0$ and $\gamma \rightarrow \infty$.

Theorem B.3: Behavior of the Softmin Gradient Function for Limit Cases

Let $x = \{x_1, \dots, x_D\} \in \mathbb{R}^D$ and $x^* = \{x_i \in x \mid x_i = \min(x)\}$. The derivative of the softmax function $\frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i}$ has the following properties for limit cases of the temperature parameter:

(a) **Hard-minimum limit:** $\lim_{\gamma \rightarrow 0} \frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \begin{cases} \frac{1}{|x^*|}, & \text{if } x_i \in x^* \\ 0, & \text{else} \end{cases}$

(b) **Mean-dominated limit:** $\lim_{\gamma \rightarrow \infty} \frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \frac{1}{|x|}$

Proofs

(a) Asymptotic Behavior of the Softmin Derivative for $\gamma \rightarrow 0$: We first write down the derivative of the softmin function w.r.t. its inputs:

$$\frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \frac{e^{-x_i/\gamma}}{\sum_{x_j \in x} e^{-x_j/\gamma}} \quad (\text{B.84})$$

and define the minimum value of the inputs $x_{\min} = \min(x)$ as well as the set of inputs that are equal to the minimum $x^* = \{x_i \in x \mid x_i = x_{\min}\}$. Then, we can reformulate (B.84) and obtain:

$$\frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \frac{e^{-(x_i - x_{\min})/\gamma}}{\sum_{x_j \in x} e^{-(x_j - x_{\min})/\gamma}} \quad (\text{B.85})$$

$$= \frac{e^{-(x_i - x_{\min})/\gamma}}{\sum_{x_j \in x^*} e^{-(x_j - x_{\min})/\gamma} + \sum_{x_j \in x \setminus x^*} e^{-(x_j - x_{\min})/\gamma}}. \quad (\text{B.86})$$

In the limit case $\gamma \rightarrow 0$, it holds that:

$$\lim_{\gamma \rightarrow 0} e^{-(x_i - x_{\min})/\gamma} = \begin{cases} 1, & \text{if } x_i \in x^* \\ 0, & \text{else,} \end{cases} \quad (\text{B.87})$$

and thus the derivative of the softmin function in the limit case becomes:

$$\lim_{\gamma \rightarrow 0} \frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \begin{cases} \frac{1}{|x^*|}, & \text{if } x_i \in x^* \\ 0, & \text{else,} \end{cases} \quad (\text{B.88})$$

which concludes the proof. If the minimum of x is unique, the gradient converges to a one-hot vector; if there are multiple minima, it converges to a uniform distribution over the minimizers. $\square$

(b) Asymptotic Behavior of the Softmin Derivative for $\gamma \rightarrow \infty$: On the other side, for $\gamma \rightarrow \infty$ we have

$$\lim_{\gamma \rightarrow \infty} e^{-(x_i - x_{\min})/\gamma} = 1 \quad \text{for } x_i \in x \quad (\text{B.89})$$

since for $x_i = x_{\min}$ the argument $x_i - x_{\min} = 0$ and for $x_i \neq x_{\min}$ the exponential term also goes to one. Thus, the derivative of the softmin function for $\gamma \rightarrow \infty$ can be approximated by a uniform distribution:

$$\lim_{\gamma \rightarrow \infty} \frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} = \frac{1}{|x|} \quad \text{for } x_i \in x, \quad (\text{B.90})$$

underscoring the mean-dominated behavior of softmin for large temperature values. $\square$

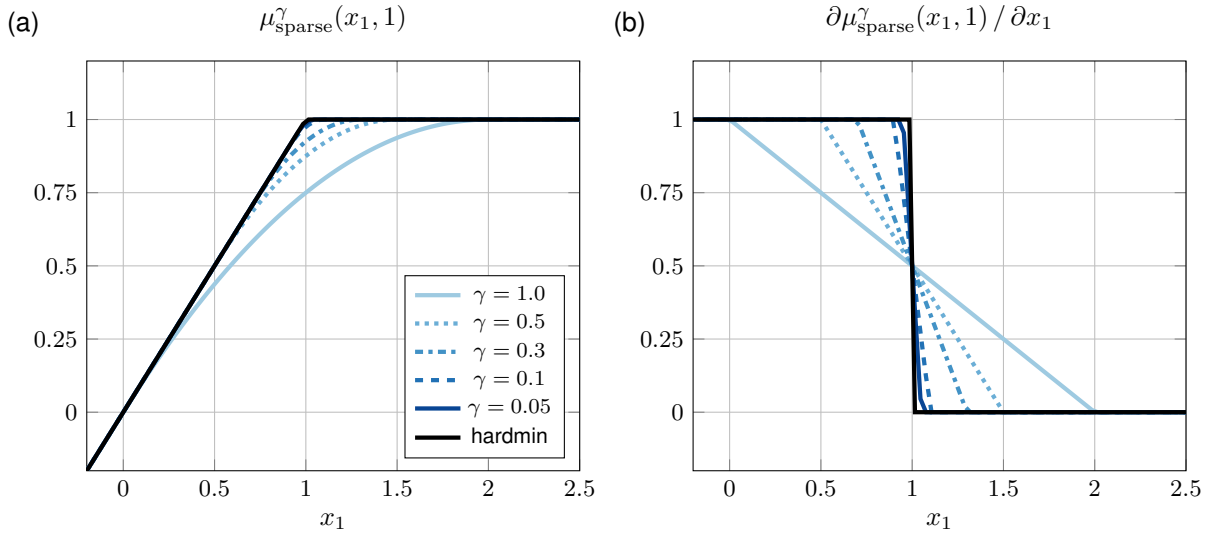


Figure B.3: Overview of (a) the sparsemin function $\mu_{\text{sparse}}^\gamma$ and (b) its partial derivative for different temperature parameters γ .

B.3 Quadratic Regularization (Sparsemin)

When the convex regularizer Ω is the negative Gini entropy, i.e., the squared ℓ_2 norm minus a constant $\Omega(q) = \frac{\gamma}{2} \langle q, q - 1 \rangle$, we obtain the *sparsemin* formulation [94] as:

$$\mu_{\text{sparse}}^\gamma(x) = \min_{q \in \Delta^D} \sum_{i=1}^D x_i q_i + \frac{\gamma}{2} \sum_{i=1}^D q_i^2 - \frac{\gamma}{2}. \quad (\text{B.91})$$

In the following section, we illustrate how the sparsemin solution can be found using a Lagrangian dual optimization approach, similar to softmin. The resulting iterative algorithm is also included in the accompanying code for [94]. The sparsemin function and its gradient are illustrated for different temperature parameters γ in Figure B.3.

To fulfill the constraint $q \in \Delta^D$, as in softmin, we again model an objective

$$f(q; x) = \sum_{i=1}^D x_i q_i + \frac{\gamma}{2} \sum_{i=1}^D q_i^2 - \frac{\gamma}{2} \quad (\text{B.92})$$

with one equality constraint

$$h(q) = \left(\sum_{i=1}^D q_i \right) - 1 = 0 \quad (\text{B.93})$$

and D inequality constraints

$$g_i(q) = -q_i \leq 0, \quad i \in [1 : D], \quad (\text{B.94})$$

associated to Lagrange multipliers $\lambda \in \mathbb{R}$ and $\mu_i \geq 0$, respectively. For the Lagrangian

$$\mathcal{L}(q, \lambda, \mu) = \sum_{i=1}^D q_i x_i + \frac{\gamma}{2} \sum_{i=1}^D q_i^2 - \frac{\gamma}{2} + \lambda h(q) + \sum_{i=1}^D \mu_i g_i(q) \quad (\text{B.95})$$

$$= \sum_{i=1}^D q_i x_i + \frac{\gamma}{2} \sum_{i=1}^D q_i^2 - \frac{\gamma}{2} - \lambda \left(1 - \sum_{i=1}^D q_i \right) - \sum_{i=1}^D \mu_i q_i, \quad (\text{B.96})$$

the KKT conditions are for all $i \in [1 : D]$:

$$\frac{\partial \mathcal{L}}{\partial q_i} = x_i + \gamma q_i + \lambda - \mu_i = 0 \quad (\text{stationarity}) \quad (\text{B.97})$$

$$g_i(q) = -q_i \leq 0 \quad (\text{primal feasibility}) \quad (\text{B.98})$$

$$h(q) = \left(\sum_{i=1}^D q_i \right) - 1 = 0 \quad (\text{primal feasibility}) \quad (\text{B.99})$$

$$\mu_i \geq 0 \quad (\text{dual feasibility}) \quad (\text{B.100})$$

$$\mu_i g_i(q) = 0 \quad (\text{complementary slackness}). \quad (\text{B.101})$$

As opposed to the derivation of the softmin formulation, we now define a set of “active” (i.e., non-zero) inequality constraints:

$$\mathcal{Q} = \{i \in [1 : D] \mid q_i > 0\}. \quad (\text{B.102})$$

In order to satisfy complementary slackness from (B.101), for all indices in the active set it must hold that:

$$\mu_i = 0 \quad \text{for } i \in \mathcal{Q}. \quad (\text{B.103})$$

Thus, the stationarity condition from (B.97) simplifies to:

$$x_i + \gamma q_i + \lambda = 0 \quad \text{for } i \in \mathcal{Q} \quad (\text{B.104})$$

$$q_i = -\frac{x_i + \lambda}{\gamma} \quad \text{for } i \in \mathcal{Q}. \quad (\text{B.105})$$

As it must hold that $q_i = 0$ for all indices $i \notin \mathcal{Q}$ not being in the set of active inequality constraints, it is sufficient to enforce the simplex constraint from (B.99) only over the active conditions $i \in \mathcal{Q}$, yielding

$$\sum_{i \in \mathcal{Q}} q_i = 1. \quad (\text{B.106})$$

Appendix

Next, we substitute (B.105) in (B.106) and obtain:

$$-\sum_{i \in Q} \frac{x_i + \lambda}{\gamma} = 1 \quad (\text{B.107})$$

$$\lambda = -\frac{\gamma + \sum_{i \in Q} x_i}{|Q|}. \quad (\text{B.108})$$

Substituting this result back into (B.105) yields:

$$q_i = \frac{1}{|Q|} \left(1 + \frac{\sum_{j \in Q} x_j}{\gamma} \right) - \frac{x_i}{\gamma} \quad \text{for } i \in Q. \quad (\text{B.109})$$

At this point, we need to ensure that indeed $q_i > 0$ for all indices $i \in Q$ in the active set, and possibly remove those indices from Q that violate the strict positivity constraint. Therefore, we define the following iterative algorithm, initialized with an active set $Q = \{1, \dots, D\}$ containing all indices:

1. compute q_i using (B.109) for all $i \in Q$
2. remove all indices i with $q_i \leq 0$ from the active set Q
3. repeat until $q_i > 0$ for all $i \in Q$.

In practice (see implementation accompanying [94]), since the largest x_i are the first ones to violate the non-negativity constraint, one sorts $\{x_i\}_{i=1}^D$ in descending order and incrementally reduces the size of the active set one by one (by removing the largest element in each iteration) until all $q_i > 0$ for $i \in Q$, yielding a worst-case runtime of $D - 1$ iterations [94]. After convergence, the optimum solution for q (and thus the gradient for $\mu_{\text{sparse}}^\gamma$) is given as

$$q_i^* = \begin{cases} \frac{1}{|Q|} \left(1 + \frac{\sum_{j \in Q} x_j}{\gamma} \right) - \frac{x_i}{\gamma} & i \in Q \\ 0 & i \notin Q. \end{cases} \quad (\text{B.110})$$

Inserting the gradient vector $q^* = \{q_i, \dots, q_D\} \in \Delta^D$ (with q_i^* calculated as in (B.110)) in (B.91) yields the final value of $\mu_{\text{sparse}}^\gamma(x)$:

$$\mu_{\text{sparse}}^\gamma(x) = \langle x, q^* \rangle + \frac{\gamma}{2} \|q^*\|_2^2 - \frac{\gamma}{2}. \quad (\text{B.111})$$

Unlike the entropic regularizer used for the softmin function (see Section B.2), the quadratic regularizer does not prevent boundary solutions for $q_i = 0$. Therefore, optimal solutions may lie on the boundary of the simplex, yielding exact zeros in q^* and thus a “sparse” solution.

B.4 Other Heuristics for Differentiable Minimum Functions: Smoothmin

Beyond the described method of obtaining differentiable approximations of the minimum function via regularization of the convex conjugate, there exist other heuristics yielding approximations of the minimum

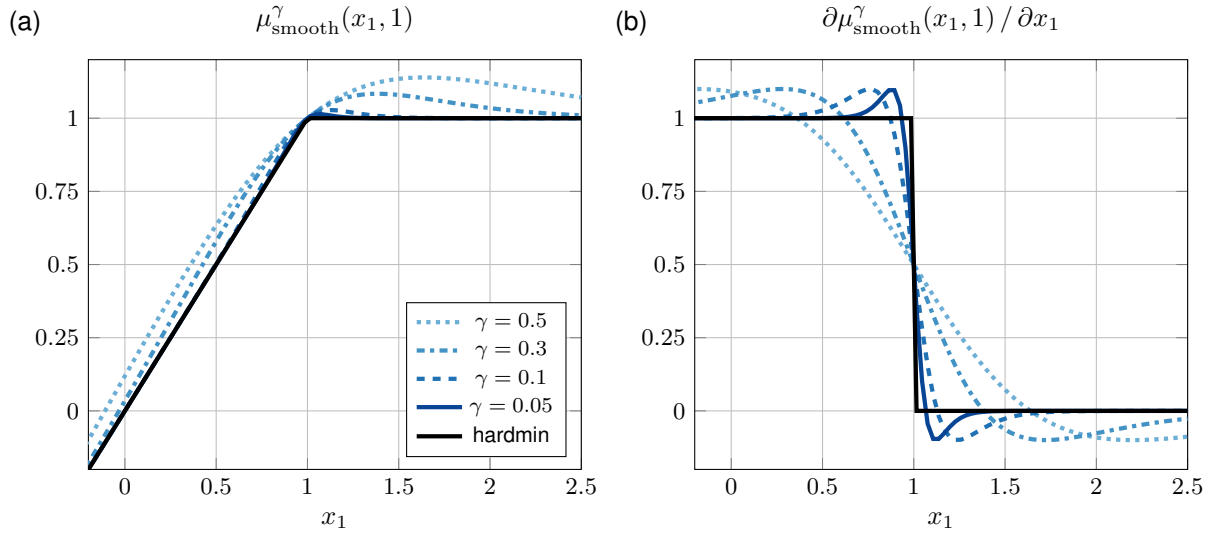


Figure B.4: Overview of (a) the smoothmin function $\mu_{\text{smooth}}^\gamma$ and (b) its partial derivative for different temperature parameters γ .

function with a continuous first-order derivative. In this section, we briefly discuss the smoothmin function [61] as an example.

The smoothmin function $\mu_{\text{smooth}}^\gamma : \mathbb{R}^D \rightarrow \mathbb{R}$ is defined via the inner product of its arguments $x = \{x_1, \dots, x_D\} \in \mathbb{R}^D$ with the softargmin function [61]:

$$\mu_{\text{smooth}}^\gamma(x) = \langle x, \nabla \mu_{\text{soft}}^\gamma(x) \rangle \quad (\text{B.112})$$

$$= \frac{\sum_{i=1}^D x_i e^{-x_i/\gamma}}{\sum_{j=1}^D e^{-x_j/\gamma}}. \quad (\text{B.113})$$

The derivative is calculated as:

$$\frac{\partial \mu_{\text{smooth}}^\gamma(x)}{\partial x_i} = \frac{e^{-x_i/\gamma}}{\sum_{j=1}^D e^{-x_j/\gamma}} \cdot \left(1 - \frac{1}{\gamma} \left(x_i - \frac{\sum_{j=1}^D x_j e^{-x_j/\gamma}}{\sum_{k=1}^D e^{-x_k/\gamma}} \right) \right) \quad (\text{B.114})$$

$$= \frac{\partial \mu_{\text{soft}}^\gamma(x)}{\partial x_i} \cdot \left(1 - \frac{1}{\gamma} (x_i - \mu_{\text{smooth}}^\gamma(x)) \right). \quad (\text{B.115})$$

As visible in Figures B.1 and B.4, the smooth minimum function exhibits several fundamental differences to softmin and sparsemin. First, while softmin and sparsemin are a lower bound to the minimum function, smoothmin provides an upper bound. Second, while the gradients of softmin and sparsemin are restricted to the probability simplex Δ^D , the derivative of smoothmin can assume negative values, values > 1 , and does not necessarily sum up to one.

C dDTW Toolbox

The following section introduces the software architecture and implementation details of the PyTorch dDTW toolbox, which was developed as part of the work on this thesis. In general, this toolbox implements the graph-based dDTW approaches presented in Chapter 10 for efficient execution on CPU and GPU, providing either full control over all dDTW parameters or parameterizations of specific variants. The toolbox is available at <https://github.com/groupmm/dDTW>.

C.1 Software Architecture

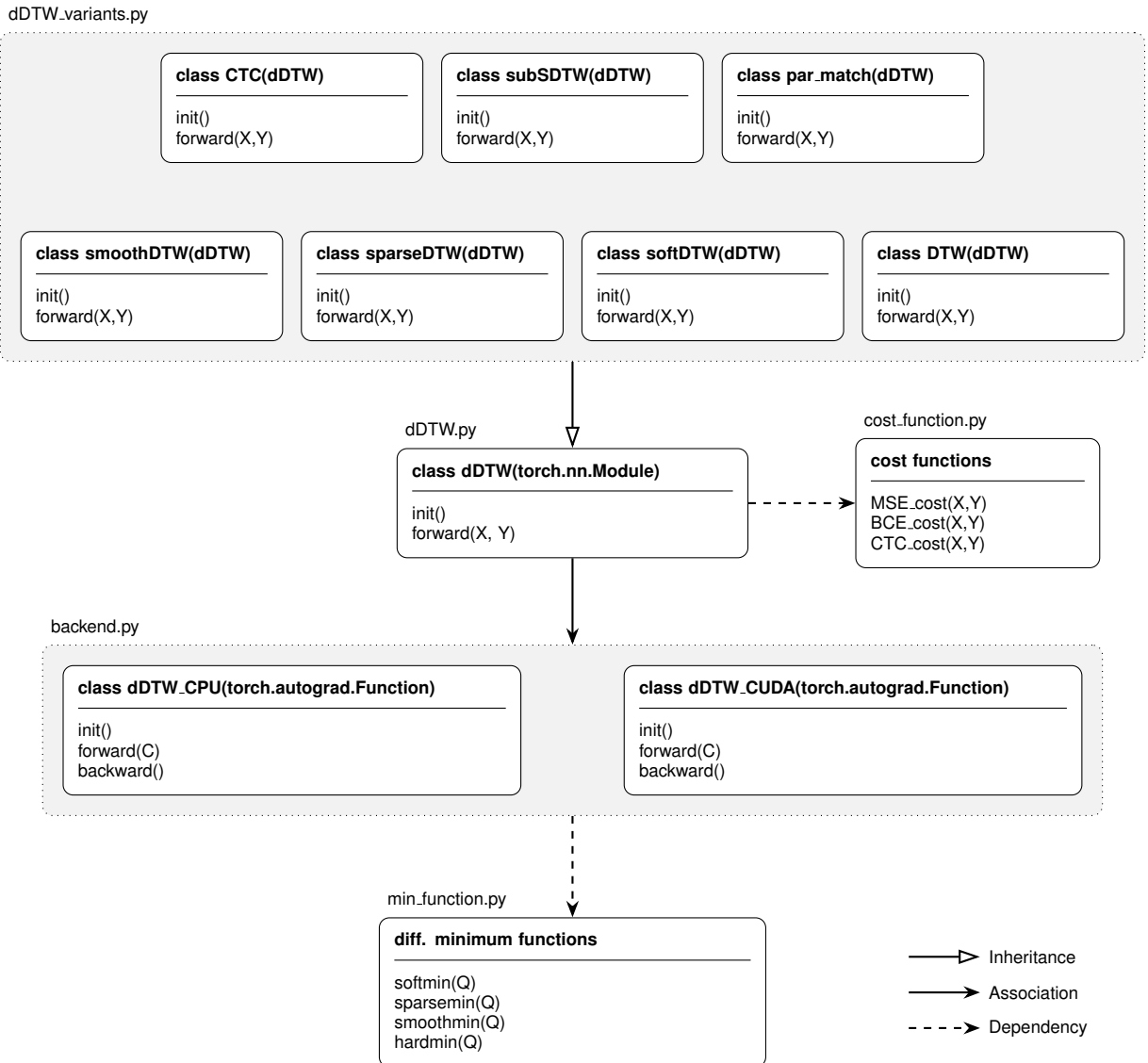


Figure C.1: Schematic overview of the dDTW toolbox software architecture.

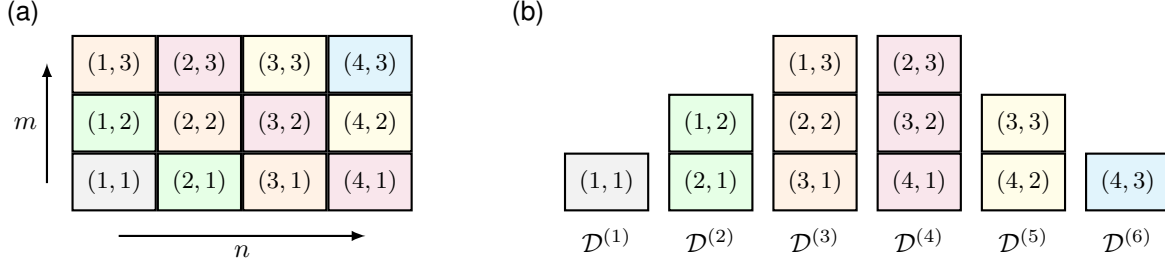


Figure C.2: Illustration of efficient DP recursion via processing along anti-diagonals for a 4×3 cost matrix. Cells with the same color are mutually independent and can be processed in parallel. (a) Cost matrix cells $\mathcal{I}$ arranged in grid form. (b) Cost matrix cells grouped according to anti-diagonals $\mathcal{D}^{(i)}$.

The dDTW toolbox software implementation (illustrated in Figure C.1) is centered around the dDTW class, which extends `torch.nn.Module` and implements a flexible differentiable alignment loss. As described in Chapter 10, the dDTW class can be parameterized with respect to the local cost function, step weights, step sizes, boundary conditions, and the choice of minimum operator. Given two input sequences X and Y , the module returns the corresponding dDTW distance.

Internally, dDTW first invokes a cost function defined in `cost_functions.py` to compute a local cost matrix $\mathbf{C}$ from the input sequences X and Y . The computation of the alignment cost itself is delegated to a backend class, either `dDTW_CPU` or `dDTW_CUDA`, depending on whether execution is performed on the CPU or GPU. These backend classes extend `torch.autograd.Function` and implement explicit forward and backward passes with efficient DP recursions. During the forward pass, intermediate states required for gradient computation are stored and reused in the backward pass. The DP recursions call a minimum operator defined in `min_functions.py`.

Although dDTW can be instantiated and parameterized directly to model arbitrary alignment graphs as specified in Sections 10.2 to 10.4, the toolbox provides a set of pre-defined variants in `dDTW_variants.py`. These classes inherit from dDTW and fix specific parameter settings, as illustrated in Section 10.5. The available variants include classical DTW, smooth DTW, sparse DTW, SDTW, subSDTW, CTC, and PM.

C.2 Efficient Implementation of DP Recursions

To increase computational speed when executing the dDTW recursions on GPU, we aim to reduce the number of originally serial computations by reformulating them in a parallel manner. The dDTW forward and backward recursions for aligning two sequences of lengths N and M are typically implemented as nested for-loops comprising $N \cdot M$ steps per forward or backward pass (see, e.g., Algorithms 10.1 and 10.2). To exploit parallelism, we group together computations corresponding to cells that are mutually independent. Specifically, as illustrated in Figure C.2a, the monotonicity constraint on the step sizes implies that all cells located on the same anti-diagonal can be computed independently [88, 172].

With the index set $\mathcal{I} := [1 : N] \times [1 : M]$, we define the set of cells $\mathcal{D}^{(i)} \subset \mathcal{I}$ on the i -th anti-diagonal as

$$\mathcal{D}^{(i)} := \{(n, m) \in \mathcal{I} \mid n + m - 1 = i\} \quad \text{for } i \in [1 : N + M - 1]. \quad (\text{C.1})$$

This construction yields a total of $N + M - 1$ anti-diagonals, each containing between 1 and $\min(N, M)$ elements, i.e.,

$$1 \leq |\mathcal{D}^{(i)}| \leq \min(N, M). \quad (\text{C.2})$$

Thus, the number of sequential computing steps reduces to $N + M - 1$, requiring at most $\min(N, M)$ parallel threads (see Figure C.2b). For the backward recursion, the mechanism of splitting the computation across anti-diagonals stays the same, only the execution order reverses. Furthermore, batch processing is possible without restrictions, as elements within a batch are also mutually independent.

List of Figures

1.1	Music alignment with DTW and SDTW.	2
2.1	Illustration of a music synchronization scenario.	10
2.2	Two-stage pipeline for music synchronization.	11
2.3	Musical score and corresponding chroma features.	12
2.4	DTW input sequences and resulting cost matrix.	14
2.5	Alignment visualization via pair-wise assignments or alignment matrix.	16
2.6	DTW forward recursion.	17
2.7	DTW backward recursion.	19
2.8	Score–audio synchronization using neural features.	20
2.9	Chroma and DLNCO features for Beethoven’s <i>Appassionata</i>	22
2.10	Heuristics for synchronization accuracy.	24
3.1	Schematic overview of the BPSD.	29
3.2	Overview of annotations in the BPSD.	42
3.3	BPSD: synchronized score–audio pairs.	45
3.4	BPSD: visualization of local key.	46
3.5	BPSD: cross-version chord recognition.	47
4.1	Illustration of the softmin function and its gradient.	51
4.2	Illustration of DTW and SDTW cost computation.	53
4.3	Illustration of DTW and SDTW gradient computation.	54
4.4	Soft alignment matrices for SDTW.	60
5.1	Stabilizing strategies for SDTW.	64
5.2	Diagonal prior on the cost matrix.	66
5.3	Strong and weak targets for the running example <i>Frühlingstraum</i>	67
5.4	SDTW alignments for different epochs and stabilizing strategies.	71
6.1	Illustration of weighted SDTW.	74
6.2	Illustration of weighted and binary alignment matrices.	78
6.3	Alignment illustration for weighted SDTW.	80

Appendix

6.4	Prediction F-measure for training with weighted SDTW.	81
6.5	Soft alignments for weighted SDTW.	82
7.1	Schematic overview of weak alignment with boundary mismatch.	86
7.2	Schematic overview of subSDTW forward and backward passes.	88
7.3	Boundary conditions for subSDTW.	92
8.1	Reformulation of SDTW into equivalent modified targets.	96
8.2	Magnitude decay in modified targets.	100
8.3	Musical score for running example.	101
8.4	Pianoroll representation of the running example.	102
8.5	Visualization of modified targets and model predictions.	103
9.1	Schematic illustration of alignment-based loss functions.	106
9.2	Overview of CTC loss computation.	107
9.3	Overview of SDTW loss computation.	112
9.4	Cost updates in the dDTW forward pass to replicate CTC.	114
9.5	Musical score of Beethoven’s piano sonata Op. 14 No. 2.	119
9.6	Single-class target representations.	119
9.7	Excerpt from “Gute Nacht” (Schubert’s “Winterreise”).	121
9.8	Multi-class target representation.	121
9.9	Predicted pitch classes for musical theme enhancement.	127
9.10	Pitch class accuracy for CTC and SDTW variants.	130
9.11	Predictions and soft alignments for CTC and SDTW variants.	131
10.1	Example of a dDTW graph.	139
10.2	dDTW graph parameterized to replicate DTW.	148
10.3	dDTW graph parameterized to replicate subDTW.	149
10.4	Example for a dDTW graph to replicate PM.	150
10.5	dDTW graph parameterized to replicate CTC.	151
A.1	Illustration of a function and its epigraph.	156
A.2	Illustration of the convexity property.	157
A.3	Illustration of the convex conjugate.	159
B.1	Differentiable minimum functions and their derivatives.	161
B.2	Softmin and its derivative for different temperature parameters.	164
B.3	Sparsemin and its derivative for different temperature parameters.	174
B.4	Smoothmin and its derivative for different temperature parameters.	177
C.1	Schematic overview of the dDTW toolbox software architecture.	179

C.2	Efficient DP recursions via parallel processing of anti-diagonals.	180
-----	--	-----

List of Tables

2.1	Short notation for synchronization components.	21
2.2	Mean synchronization accuracy for ASAP.	26
3.1	Overview of the first movements of Beethoven’s 32 Piano Sonatas.	31
3.2	Overview of the folder structure of the BPSD.	33
3.3	Overview of audio recordings in the BPSD.	36
3.4	Mean synchronization accuracy for BPSD.	37
3.5	Synchronization accuracy for BPSD audio versions.	39
5.1	Musically motivated CNN architecture.	68
5.2	Test results for SDTW and stabilizing strategies.	69
7.1	Test results for subSDTW.	94
9.1	CTC and SDTW configurations within the dDTW framework.	123
9.2	CNN architecture for musical theme enhancement.	125
9.3	Experimental results for monophonic pitch-class enhancement.	126
9.4	Experimental results for symbol-level decoding on MTD.	128
9.5	CNN architecture for pitch class estimation.	129
9.6	Runtime and memory consumption for SDTW, dDTW, and CTC.	133
10.1	Overview of dDTW parameterizations leading to different DTW variants.	147

List of Acronyms

AMT	Automatic Music Transcription	HCQT	Harmonic Constant-Q Transform
ASR	Automatic Speech Recognition	KKT	Karush–Kuhn–Tucker
BPSD	Beethoven Piano Sonata Dataset	LER	Label Error Rate
BCE	Binary Cross-Entropy	MCTC	Multi-Label Connectionist Temporal Classification
CTC	Connectionist Temporal Classification	MIDI	Musical Instrument Digital Interface
CNN	Convolutional Neural Network	MIR	Music Information Retrieval
CQT	Constant-Q Transform	MPE	Multi-Pitch Estimation
DAG	Directed Acyclic Graph	MRR	Mean Reciprocal Rank
DL	Deep Learning	MSE	Mean Squared Error
DLNCO	Decaying Locally Normalized Chroma Onset	MTD	Musical Theme Dataset
dDTW	Differentiable Dynamic Time Warping	PCE	Pitch Class Estimation
DNN	Deep Neural Network	PM	Partial Matching
DP	Dynamic Programming	SDTW	Soft Dynamic Time Warping
DTW	Dynamic Time Warping	STFT	Short-Time Fourier Transform
EM	Expectation–Maximization	subDTW	Subsequence DTW
		subSDTW	Subsequence SDTW
		SWD	Schubert Winterreise Dataset

List of Symbols

Latin Symbols

A	Alignment matrix $\in \mathbb{R}^{N \times M}$
$\mathcal{A}$	Set of valid alignment matrices
A	CTC alphabet
A'	Extended CTC alphabet
b	Cost at end boundary conditions $\in \mathbb{R}$
B	Backtracking pointers $\in \mathbb{R}^{N \times M \times (S+1)}$
$\mathcal{B}_{\text{start}}$	Start boundary conditions $\subseteq \mathcal{I}$
$\mathcal{B}_{\text{end}}$	End boundary conditions $\subseteq \mathcal{I}$
C	Local cost matrix $\in \mathbb{R}^{N \times M}$
c_{BCE}	Binary cross-entropy cost function $\mathbb{R}^D \rightarrow \mathbb{R}$
c_{MSE}	Mean squared error cost function $\mathbb{R}^D \rightarrow \mathbb{R}$
D	Accumulated cost matrix $\in \mathbb{R}^{N \times M}$
E	Set of edges in a graph
E	Soft alignment matrix $\in [0, 1]^{N \times M}$
f	Cost of incoming alignment steps $\in \mathbb{R}^{S+1}$
G	Graph
G	Gradient correction matrix $\in \mathbb{R}^{N \times M}$
H	Gradient matrix $\in \mathbb{R}^{N \times M}$
I	Index set for the input of the softmax function
$\mathcal{I}$	Set of all cost matrix cells $[1 : N] \times [1 : M]$
P	Alignment path $(p_1, \dots, p_L)$
M	Length of label sequence $\in \mathbb{N}$
N	Length of prediction sequence $\in \mathbb{N}$
$\mathcal{P}$	Set of alignment paths
S	Set of alignment steps $\{(i_1, j_s), \dots, (i_S, j_S)\}$
V	Set of vertices in a graph
W	Step weight matrix $\in \mathbb{R}^{N \times M \times S}$
w_{step}	Local step weight $\in \mathbb{R}$
w_{start}	Path start weight $\in \mathbb{R}$
w_{end}	Path end weight $\in \mathbb{R}$
X	Predicted sequence $(x_1, \dots, x_N)$
Y	Label sequence $(y_1, \dots, y_M)$

Greek Symbols

γ	Temperature parameter for diff. min. functions $\in \mathbb{R}$
Δ^D	D -dimensional probability simplex $\subset [0, 1]^D$
$\mathcal{E}^D$	Set of standard basis vectors in $\mathbb{R}^D$
ζ	Binarization operator for weighted alignment matrices $\mathbb{R}_{\geq 0}^{N \times M} \rightarrow \{0, 1\}^{N \times M}$
θ	Edge weight in a graph $\in \mathbb{R}$
Θ	Set of all edge weights in a graph
μ	General minimum function $\mathbb{R}^I \rightarrow \mathbb{R}$
μ_{soft}^γ	Softmin function $\mathbb{R}^I \rightarrow \mathbb{R}$
π	CTC label path $\in A'^N$
Ω	Convex regularizer $\mathbb{R}^D \rightarrow \mathbb{R}$

Mathematical Notation

$\{a, b, c\}$	Set
(a, a, b, c, a)	Sequence
$[\alpha, \beta]$	Continuous range $\{x \in \mathbb{R} \mid \alpha \leq x \leq \beta\}$
$[1 : N]$	Discrete integer range $\{1, \dots, N\}$
$\langle \cdot, \cdot \rangle$	Frobenius inner product
f^*	Convex conjugate of a function f

Bibliography

- [1] Ruchit Agrawal, Daniel Wolff, and Simon Dixon. A convolutional-attentional neural framework for structure-aware performance-score synchronization. *IEEE Signal Processing Letters*, 29:344–348, 2021. doi: 10.1109/LSP.2021.3135192.
- [2] Vlora Arifi, Michael Clausen, Frank Kurth, and Meinard Müller. Synchronization of music data in score-, MIDI- and PCM-format. *Computing in Musicology*, 13:9–33, 2004.
- [3] Andreas Arzt, Gerhard Widmer, and Simon Dixon. Automatic page turning for musicians via real-time machine listening. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 241–245, Amsterdam, The Netherlands, 2008. IOS Press. doi: 10.3233/978-1-58603-891-5-241.
- [4] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, Virtual, 2020.
- [5] 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.
- [6] 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.
- [7] Stefan Balke, Johannes Zeitler, Vlora Arifi-Müller, Brian McFee, Tomoyasu Nakano, Masataka Goto, and Meinard Müller. RWC revisited: Towards a community-driven MIR corpus. *Transactions of the International Society for Music Information Retrieval*, 9:21–35, 2026. doi: 10.5334/tismir.326.
- [8] Cyril Banderier and Sylviane Schwer. Why delannoy numbers? *Journal of Statistical Planning and Inference*, 135:40–54, 2005. doi: 10.1016/j.jspi.2005.02.004.
- [9] 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.
- [10] 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.
- [11] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952. doi: 10.1073/pnas.38.8.7.

- [12] 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.
- [13] 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.
- [14] 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.
- [15] Rachel M. Bittner, Magdalena Fuentes, David Rubinstein, Andreas Jansson, Keunwoo Choi, and Thor Kell. mirdata: Software for reproducible usage of datasets. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 99–106, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527749.
- [16] Mathieu Blondel and Vincent Roulet. The elements of differentiable programming. *CoRR*, abs/2403.14606, 2024. doi: 10.48550/arXiv.2403.14606.
- [17] Mathieu Blondel, Arthur Mensch, and Jean-Philippe Vert. Differentiable divergences between time series. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3853–3861, Virtual, 2021. URL <http://proceedings.mlr.press/v130/blondel121a.html>.
- [18] Sebastian Böck, Andreas Arzt, Florian Krebs, and Markus Schedl. Online real-time onset detection with recurrent neural networks. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, York, UK, September 2012. doi: DOI:10.13140/2.1.3067.2649.
- [19] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 255–261, New York City, New York, USA, 2016. doi: 10.5281/zenodo.1415835.
- [20] Sebastian Böck, Matthew E. P. Davies, and Peter Knees. Multi-task learning of tempo and beat: Learning one to improve the other. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 486–493, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527849.
- [21] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0521833783.
- [22] Irmak Bukey, Jason Zhang, and TJ Tsai. FlexDTW: Dynamic time warping with flexible boundary conditions. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 733–740, Milano, Italy, 2023. doi: 10.5281/ZENODO.10265393.
- [23] Jorge Calvo-Zaragoza, Jan Hajič Jr., and Alexander Pacha. Understanding optical music recognition. *ACM Computing Surveys*, 53(4), 2020. doi: 10.1145/3397499.
- [24] Chris Cannam, Christian Landone, and Mark B. Sandler. Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the International Conference on Multimedia*, pages 1467–1468, Florence, Italy, 2010. doi: 10.1145/1873951.1874248.

- [25] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Nieves. D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3546–3555, Long Beach, CA, USA, 2019. doi: 10.1109/CVPR.2019.00366.
- [26] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 90–97, Paris, France, 2018. doi: 10.5281/zenodo.1492350.
- [27] Ching-Yu Chiu, Johannes Zeitler, Vlora Arifi-Müller, and Meinard Müller. Downbeat tracking for Western classical music recordings: A case study for Beethoven piano sonatas. In *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, pages 1–4, Hannover, Germany, 2024.
- [28] Barry Cooper. *The creation of Beethoven’s 35 piano sonatas*. Routledge, London, UK, 2017. ISBN 9781315615080. doi: 10.4324/9781315615080.
- [29] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, volume 2, pages 2292—2300, 2013.
- [30] Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 894–903, Sydney, NSW, Australia, 2017. URL <http://proceedings.mlr.press/v70/cuturi17a.html>.
- [31] David Damm, Christian Fremerey, Verena Thomas, Michael Clausen, Frank Kurth, and Meinard Müller. A digital library framework for heterogeneous music collections: From document acquisition to cross-modal interaction. *International Journal on Digital Libraries: Special Issue on Music Digital Libraries*, 12(2-3): 53–71, 2012. doi: 10.1007/s00799-012-0087-y.
- [32] David Damschroder. *Harmony in Beethoven*. Cambridge University Press, 2016. ISBN 9781316471043. doi: 10.1017/CBO9781316471043.
- [33] Roger B. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 193–198, Paris, France, 1984.
- [34] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271, 1959. doi: 10.1007/BF01386390.
- [35] Christian Dittmar, Bernhard Lehner, Thomas Prätzlich, Meinard Müller, and Gerhard Widmer. Cross-version singing voice detection in classical opera recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 618–624, Málaga, Spain, October 2015. doi: 10.5281/zenodo.1416958.
- [36] Christian Dittmar, Johannes Zeitler, Stefan Balke, Simon Schwär, and Meinard Müller. PULSE-IT: Lightweight and expressive synthesis of wind instrument playing. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Daejeon, South Korea, 2025.

- [37] 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.
- [38] 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.
- [39] Nikita Dvornik, Isma Hadji, Konstantinos G. Derpanis, Animesh Garg, and Allan D. Jepson. Drop-dtw: Aligning common signal between sequences while dropping outliers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13782–13793, Virtual, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/729c68884bd359ade15d5f163166738a-Abstract.html>.
- [40] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1801–1810, Long Beach, CA, USA, 2019. doi: 10.1109/CVPR.2019.00190.
- [41] Drew Edwards, Simon Dixon, Emmanouil Benetos, Akira Maezawa, and Yuta Kusaka. A data-driven analysis of robust automatic piano transcription. *IEEE Signal Processing Letters*, 31:681–685, 2024. doi: 10.1109/LSP.2024.3363646.
- [42] 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.
- [43] Jesse Engel, Rigel Swavely, Lamtharn Hanoi Hantrakul, Adam Roberts, and Curtis Hawthorne. Self-supervised pitch detection by inverse audio synthesis. In *International Conference on Machine Learning (ICML), Workshop on Self-Supervision in Audio and Speech*, Vienna, Austria, 2020.
- [44] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, 2009. doi: 10.1109/ICASSP.2009.4959972.
- [45] Sebastian Ewert, Meinard Müller, Verena Konz, Daniel Müllensiefen, and Gerraint A. Wiggins. Towards cross-version harmonic analysis of music. *IEEE Transactions on Multimedia*, 14(3-2):770–782, 2012. doi: 10.1109/TMM.2012.2190047.
- [46] Sebastian Ewert, Bryan Pardo, Meinard Müller, and Mark Plumbley. Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3):116–124, April 2014. doi: 10.1109/MSP.2013.2296076.
- [47] Francesco Foscarin, Andrew McLeod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. ASAP: A dataset of aligned scores and performances for piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 534–541, 2020. doi: 10.5281/zenodo.4245489.
- [48] Thassilo Gadermaier and Gerhard Widmer. A study of annotation and alignment accuracy for performance comparison in complex orchestral music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 769–775, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527923.

- [49] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse H. Engel. MT3: multi-task multitrack music transcription. *Computing Research Repository (CoRR)*, abs/2111.03017, 2021. URL <https://arxiv.org/abs/2111.03017>.
- [50] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse H. Engel. MT3: Multi-task multitrack music transcription. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Virtual, 2022.
- [51] Beat Gfeller, Christian Frank, Dominik Roblek, Matthew Sharifi, Marco Tagliasacchi, and Mihajlo Velimirovic. SPICE: Self-supervised pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1118–1128, 2020. doi: 10.1109/TASLP.2020.2982285.
- [52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge and London, 2016. ISBN 978-0-262-03561-3.
- [53] Mark Gotham and Matthew Ireland. Taking form: A representation standard, conversion code, and example corpora for recording, visualizing, and studying analyses of musical form. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 693–699, Delft, The Netherlands, 2019. doi: 10.5281/zenodo.3527903.
- [54] Mark Gotham, Johannes Hentschel, Louis Couturier, Nathan Dykeaylen, Martin Rohrmeier, and Mathieu Giraud. The ‘Measure Map’: an inter-operable standard for aligning symbolic music. In *Proceedings of the International Conference on Digital Libraries for Musicology (DLfM)*, pages 91–99, Milano, Italy, 2023. doi: 10.1145/3625135.3625136.
- [55] Mark Gotham, Gianluca Micchi, Néstor Nápoles López, and Malcolm Sailor. When in rome: A meta-corpus of functional harmony. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 6 (1):150–166, 2023. doi: 10.5334/TISMIR.165.
- [56] Masataka Goto. Development of the RWC music database. In *Proceedings of the International Congress on Acoustics (ICA)*, pages 553–556, 2004.
- [57] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. doi: 10.48550/arXiv.1211.3711.
- [58] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1764–1772, Beijing, China, 2014.
- [59] 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.
- [60] Chitrallekha 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.

- [61] Isma Hadji, Konstantinos G. Derpanis, and Allan D. Jepson. Representation learning via global temporal alignment and cycle-consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11068–11077, Virtual, 2021. doi: 10.1109/CVPR46437.2021.01092.
- [62] Christopher Harte, Mark B. Sandler, Samer Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 66–71, London, UK, 2005. doi: 10.5281/zenodo.1415113.
- [63] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse H. Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference, (ISMIR)*, pages 50–57, 2018. doi: 10.5281/zenodo.1492341.
- [64] 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. URL <https://openreview.net/forum?id=r11YRjC9F7>.
- [65] Curtis Hawthorne, Ian Simon, Rigel Swavely, Ethan Manilow, and Jesse H. Engel. Sequence-to-sequence piano transcription with transformers. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 246–253, Online, 2021. doi: 10.5281/zenodo.5624460.
- [66] Johannes Hentschel, Markus Neuwirth, and Martin Rohrmeier. The annotated Mozart sonatas: Score, harmony, and cadence. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 4 (1):67–80, 2021. doi: 10.5334/TISMIR.63.
- [67] Johannes Hentschel, Yannis Rammos, Fabian C. Moss, Markus Neuwirth, and Martin Rohrmeier. An annotated corpus of tonal piano music from the long 19th century. *Empirical Musicology Review*, 18(1): 84–95, 2023. doi: 10.18061/emr.v18i1.8903.
- [68] James Hepokoski and Warren Darcy. *Elements of Sonata Theory. Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata*. Oxford University Press, 2006. doi: 10.1093/acprof:oso/9780195146400.001.0001.
- [69] Yo-Wei Hsiao, Tzu-Yun Hung, Tsung-Ping Chen, and Li Su. BPS-Motif: A dataset for repeated pattern discovery of polyphonic symbolic music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 281–288, 2023. doi: 10.5281/ZENODO.10265277.
- [70] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975. doi: 10.1109/TASSP.1975.1162641.
- [71] Nanzhu Jiang and Meinard Müller. Automated methods for analyzing music recordings in sonata form. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 595–600, Curitiba, Brazil, 2013. doi: 10.5281/zenodo.1418282.
- [72] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, New York City, New York, USA, 2016. doi: 10.5281/zenodo.1416488.

- [73] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. CREPE: A convolutional representation for pitch estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 161–165, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8461329.
- [74] 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. URL <http://arxiv.org/abs/1412.6980>.
- [75] Stephanie Klauk, Reiner Kleinertz, Christof Weiß, and Meinard Müller. Seitensatz versus Mittelsatz: Expositionen in Beethovens frühen Klaviersonaten zwischen zeitgenössischer Theorie und computergestützter Analyse. In Simone Hohmaier, editor, *Jahrbuch 2017 des Staatlichen Instituts für Musikforschung (SIM) – Preußischer Kulturbesitz*, pages 271–300. Schott Music, 2021. ISBN 978-3-7957-1005-7.
- [76] Rainer Kleinertz. Streichquartette von Joseph Haydn und Wolfgang Amadeus Mozart im Spiegel der Sonatentheorie Francesco Galeazzis. In Christian Speck, editor, *The String Quartet from 1750 to 1870: From the Private to the Public Sphere*, volume 26 of *Speculum Musicae*, pages 297–316. Brepols, 2016.
- [77] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions of Audio, Speech, and Language Processing*, 29:3707–3717, 2021. doi: 10.1109/TASLP.2021.3121991.
- [78] Verena Konz, Meinard Müller, and Rainer Kleinertz. A cross-version chord labelling approach for exploring harmonic structures—a case study on Beethoven’s Appassionata. *Journal of New Music Research*, 42(1): 61–77, 2013. doi: 10.1080/09298215.2012.750369.
- [79] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag, 5th edition, 2012. ISBN 3642244874.
- [80] 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.
- [81] 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.
- [82] Michael Krause, Sebastian Strahl, and Meinard Müller. Weakly supervised multi-pitch estimation using cross-version alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Milano, Italy, 2023. doi: 10.5281/zenodo.10265278.
- [83] Michael Krause, Christof Weiß, and Meinard Müller. Soft dynamic time warping for multi-pitch estimation and beyond. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1–5, Rhodes Island, Greece, 2023. doi: 10.1109/ICASSP49357.2023.10095907.
- [84] Frank Kurth, Meinard Müller, David Damm, Christian Fremerey, Andreas Ribbrock, and Michael Clausen. SyncPlayer—an advanced system for multimodal music access. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 381–388, London, UK, November 2005. doi: 10.5281/zenodo.1416495.

Appendix

- [85] Alexander Lerch, Claire Arthur, Ashis Pati, and Siddharth Gururani. An interdisciplinary review of music performance analysis. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3 (1):221–245, 2020. doi: 10.5334/tismir.53.
- [86] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [87] Robert Macrae and Simon Dixon. Accurate real-time windowed time warping. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 423–428, Utrecht, The Netherlands, 2010. doi: 10.5281/zenodo.1416155.
- [88] Mehran Maghoughi, Eugene Matthew Taranta, and Joseph LaViola. DeepNAG: Deep non-adversarial gesture generation. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 213–223, College Station, Texas, USA, 2021. doi: 10.1145/3397481.3450675.
- [89] Ben Maman. *Enhancing Music Information Retrieval and Human–Computer Interaction with Machine Learning*. PhD thesis, Tel Aviv University, Tel Aviv, Israel, 2024.
- [90] Ben Maman and Amit H. Bermano. Unaligned supervision for automatic music transcription in the wild. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 14918–14934, Baltimore, Maryland, USA, 2022.
- [91] Ben Maman, Johannes Zeitler, Meinard Müller, and Amit H. Bermano. Performance conditioning for diffusion-based multi-instrument music synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5045–5049, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10445979.
- [92] Ben Maman, Johannes Zeitler, Meinard Müller, and Amit H. Bermano. Multi-aspect conditioning for diffusion-based music synthesis: Enhancing realism and acoustic control. *IEEE Transactions on Audio, Speech, and Language Processing*, 33:68–81, 2025. doi: 10.1109/TASLP.2024.3507553.
- [93] Matthias Mauch, Robert M. MacCallum, Mark Levy, and Armand M. Leroi. The evolution of popular music: USA 1960–2010. *Royal Society Open Science*, 2(5), 2015. doi: 10.1098/rsos.150081.
- [94] Arthur Mensch and Mathieu Blondel. Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3459–3468, Stockholmsmässan, Stockholm, Sweden, 2018. URL <http://proceedings.mlr.press/v80/mensch18a.html>.
- [95] David Meredith, editor. *Computational Music Analysis*. Springer, 2016. ISBN 978-3-319-25929-1. doi: 10.1007/978-3-319-25931-4.
- [96] Fabian C. Moss, Markus Neuwirth, Daniel Harasim, and Martin Rohrmeier. Statistical characteristics of tonal harmony: A corpus study of beethoven’s string quartets. *PLOS ONE*, 14(6), 2019. doi: 10.1371/journal.pone.0217242.
- [97] Meinard Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007. doi: 10.1007/978-3-540-74048-3.

- [98] Meinard Müller. *Fundamentals of Music Processing – Using Python and Jupyter Notebooks*. Springer Verlag, 2nd edition, 2021. ISBN 978-3-030-69807-2. doi: 10.1007/978-3-030-69808-9.
- [99] Meinard Müller and Daniel Appelt. Path-constrained partial music synchronization. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 65–68, Las Vegas, Nevada, USA, April 2008. doi: 10.1109/ICASSP.2008.4517547.
- [100] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 192–197, Victoria, Canada, October 2006. doi: 10.5281/zenodo.1417408.
- [101] Meinard Müller, Verena Konz, Wolfgang Bogler, and Vlori 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.
- [102] 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.
- [103] Meinard Müller, Yigitcan Özer, Michael Krause, Thomas Prätzlich, and Jonathan Driedger. Sync Toolbox: A Python package for efficient, robust, and accurate music synchronization. *Journal of Open Source Software (JOSS)*, 6(64):3434:1–4, 2021. doi: 10.21105/joss.03434.
- [104] Meinard Müller, Johannes Zeitler, and Christof Weiß. A computational approach for visualizing tonal structures in music recordings. In *Oxford Handbook for Systematic Empirical Research in the Arts*, Oxford Handbooks. Oxford University Press, accepted 2025.
- [105] Meinard Müller, Johannes Zeitler, and Sebastian Strahl. PCPT notebooks: A preparation course for PyTorch with a focus on signal processing. *Journal of Open Source Education (JOSE)*, submitted 2025.
- [106] Cory Myers, Lawrence Rabiner, and Aaron Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980. doi: 10.1109/TASSP.1980.1163491.
- [107] Eita Nakamura and Kunihiko Kaneko. Statistical evolutionary laws in music styles. *Scientific Reports*, 9(1), 2019. doi: 10.1038/s41598-019-52380-6.
- [108] Markus Neuwirth. A myriad of exceptions? Challenging the sonata-form typology for the classical repertoire. *Music Analysis*, 40(3):312–352, 2021. doi: doi.org/10.1111/musa.12186.
- [109] Markus Neuwirth, Daniel Harasim, Fabian C. Moss, and Martin Rohrmeier. The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets. *Frontiers in Digital Humanities*, 5: 16, 2018. doi: 10.3389/fdigh.2018.00016.
- [110] Bernhard Niedermayer and Gerhard Widmer. A multi-pass algorithm for accurate audio-to-score alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 417–422, Utrecht, The Netherlands, 2010. doi: 10.5281/zenodo.1415909.

- [111] Oriol Nieto, Gautham J. Mysore, Cheng-i Wang, Jordan B. L. Smith, Jan Schlüter, Thomas Grill, and Brian McFee. Audio-based music structure analysis: Current trends, open challenges, and applications. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 3(1):246–263, 2020. doi: 10.5334/tismir.54.
- [112] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer (Springer Series in Operations Research and Financial Engineering), 2006. doi: 10.1007/978-0-387-40065-5.
- [113] Yigitcan Özer, Matěj Ištvánek, Vlora Arifi-Müller, and Meinard Müller. Using activation functions for improving measure-level audio synchronization. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 749–756, Bengaluru, India, 2022. doi: 10.5281/zenodo.7316771.
- [114] Yigitcan Özer, Simon Schwär, Vlora Arifi-Müller, Jeremy Lawrence, Emre Sen, and Meinard Müller. Piano Concerto Dataset (PCD): A multitrack dataset of piano concertos. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 6(1):75–88, 2023. doi: 10.5334/tismir.160.
- [115] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, Vancouver, BC, Canada, 2019.
- [116] 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.
- [117] Silvan David Peter, Email Silvan David Peter, Carlos Eduardo Cancino-Chacó, Francesco Foscari, Andrew Philip McLeod, Florian Henkel, Emmanouil Karystinaios, and Gerhard Widmer. Automatic note-level score-to-performance alignments in the ASAP dataset. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 6(1):27–42, 2023. doi: 10.5334/tismir.149.
- [118] Pavel A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, 2000. ISBN 9780262281522. doi: 10.7551/mitpress/2022.001.0001.
- [119] Thomas Prätzlich and Meinard Müller. Triple-based analysis of music alignments without the need of ground-truth annotations. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 266–270, Shanghai, China, March 2016. doi: 10.1109/ICASSP.2016.7471678.
- [120] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, March 2016. doi: 10.1109/ICASSP.2016.7471739.
- [121] Xavier Riley, Drew Edwards, and Simon Dixon. High resolution guitar transcription via domain adaptation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1051–1055, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10446182.

- [122] Alain Riou, Stefan Lattner, Gaëtan Hadjeres, and Geoffroy Peeters. PESTO: Pitch estimation with self-supervised transposition-equivariant objective. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 535–544, Milano, Italy, 2023. doi: 10.5281/zenodo.10265342.
- [123] R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1998. ISBN 978-3-540-62772-2. doi: 10.1007/978-3-642-02431-3.
- [124] 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.
- [125] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. doi: 10.1109/TASSP.1978.1163055.
- [126] Stan Salvador and Philip Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Proceedings of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.
- [127] Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6979–6983, Florence, Italy, May 2014. doi: 10.1109/ICASSP.2014.6854953.
- [128] Pascal Schmolenzky, Stephanie Klauk, Rainer Kleinertz, Christof Weiß, and Meinard Müller. A multidimensional approach to opera analysis: Harmony, tempo, and dramatic interaction in wagner’s siegfried. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 142–149, Daejeon, South Korea, 2025. doi: 10.5281/ZENODO.17811336.
- [129] 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.
- [130] Diemo Schwarz, Nicola Orio, and Norbert Schnell. Robust polyphonic midi score following with hidden Markov models. In *International Computer Music Conference (ICMC)*, Miami, Florida, USA, 2004.
- [131] 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.
- [132] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2017. doi: 10.1109/TPAMI.2016.2646371.
- [133] Kevin Shih, Rafael Valle, Rohan Badlani, Adrian Łańcucki, Wei Piang, and Bryan Catanzaro. RAD-TTS: Parallel flow-based TTS with robust alignment learning and diverse synthesis. In *International Conference on Machine Learning (ICML), Third Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, Virtual, 2021. URL <https://openreview.net/pdf?id=0NQwnnwAORi>.

- [134] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981. doi: 10.1016/0022-2836(81)90087-5.
- [135] 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.
- [136] 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. URL <https://openreview.net/forum?id=rkFBJv9gg>.
- [137] John Thickstun, Zaïd Harchaoui, Dean P. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245, Calgary, Canada, 2018. doi: 10.1109/ICASSP.2018.8461686.
- [138] Bernardo Torres, Geoffroy Peeters, and Gaël Richard. Unsupervised harmonic parameter estimation using differentiable DSP and spectral optimal transport. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1176–1180, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10447011.
- [139] Bernardo Torres, Alain Riou, Gaël Richard, and Geoffroy Peeters. Translation-equivariant self-supervised learning for pitch estimation with optimal transport. In *Late-Breaking and Demo Session of the International Conference on Music Information Retrieval (ISMIR)*, pages 1–3, Daejeon, South Korea, 2025.
- [140] Donald Francis Tovey. *A Companion to Beethoven’s Pianoforte Sonatas*. The Associated Board of the Royal Schools of Music, 1998.
- [141] George Trigeorgis, Mihalis A. Nicolaou, Björ W. Schuller, and Stefanos Zafeiriou. Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1128–1138, 2018. doi: 10.1109/TPAMI.2017.2710047.
- [142] 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.
- [143] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, Long Beach, CA, USA, 2017.
- [144] Barry Vercoe. The synthetic performer in the context of live performance. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 199–200, 1984.
- [145] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010.

- [146] Christof Weiß and Geoffroy Peeters. Training deep pitch-class representations with a multi-label CTC loss. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 754–761, Online, 2021. doi: 10.5281/zenodo.5624358.
- [147] Christof Weiß and Geoffroy Peeters. Learning multi-pitch estimation from weakly aligned score-audio pairs using a multi-label CTC loss. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 121–125, New Paltz, USA, 2021. doi: 10.1109/WASPAA52581.2021.9632740.
- [148] Christof Weiß and Geoffroy Peeters. Comparing deep models and evaluation strategies for multi-pitch estimation in music recordings. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 30: 2814–2827, 2022. doi: 10.1109/TASLP.2022.3200547.
- [149] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for Western classical music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 517–523, New York, USA, 2016. doi: 10.5281/zenodo.1417449.
- [150] 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.
- [151] Christof Weiß, Stephanie Klauk, Mark Gotham, Meinard Müller, and Rainer Kleinertz. Discourse not dualism: An interdisciplinary dialogue on sonata form in Beethoven’s early piano sonatas. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 199–206, Montréal, Canada, 2020. doi: 10.5281/zenodo.4245401.
- [152] Christof Weiß, Hendrik Schreiber, and Meinard Müller. Local key estimation in music recordings: A case study across songs, versions, and annotators. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2919–2932, 2020. doi: 10.1109/TASLP.2020.3030485.
- [153] Christof Weiß, Frank Zalkow, Vlora Arifi-Müller, Meinard Müller, Hendrik Vincent Kooops, Anja Volk, and Harald Grohgan. Schubert Winterreise dataset: A multimodal scenario for music analysis. *ACM Journal on Computing and Cultural Heritage (JOCCH)*, 14(2):25:1–18, 2021. doi: 10.1145/3429743.
- [154] Christof Weiß, Johannes Zeitler, Tim Zunner, Florian Schuberth, and Meinard Müller. Learning pitch-class representations from score–audio pairs of classical music. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 746–753, Online, 2021. doi: 10.5281/zenodo.5624549.
- [155] Christof Weiß, Vlora Arifi-Müller, Michael Krause, Frank Zalkow, Stephanie Klauk, Rainer Kleinertz, and Meinard Müller. Wagner Ring Dataset: A complex opera scenario for music processing and computational musicology. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 6(1): 135–149, 2023. doi: 10.5334/tismir.161.
- [156] Curtis Wigington, Brian L. Price, and Scott Cohen. Multi-label connectionist temporal classification. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 979–986, Sydney, Australia, 2019. doi: 10.1109/ICDAR.2019.00161.

- [157] 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, and Language Processing*, 27(2): 355–366, 2019. doi: 10.1109/TASLP.2018.2879399.
- [158] Yu-Te Wu, Berlin Chen, and Li Su. Polyphonic music transcription with semantic segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 166–170, Brighton, UK, 2019. doi: 10.1109/ICASSP.2019.8682605.
- [159] Jonathan Yaffe, Ben Maman, Meinard Müller, and Amit Bermano. Count the notes: Histogram-based supervision for automatic music transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 469–476, Daejeon, South Korea, 2025. doi: 10.5281/ZENODO.17706488.
- [160] Frank Zalkow and Meinard Müller. CTC-based learning of chroma features for score-audio music retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2957–2971, 2021. doi: 10.1109/TASLP.2021.3110137.
- [161] Frank Zalkow, Sebastian Rosenzweig, Johannes Graulich, Lukas Dietz, El Mehdi Lemnaouar, and Meinard Müller. A web-based interface for score following and track switching in choral music. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.
- [162] 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.
- [163] Eberhard Zeidler. *Springer-Taschenbuch der Mathematik. Begründet von I.N. Bronstein und K.A. Semendjaew Weitergeführt von G. Grosche, V. Ziegler und D. Ziegler Herausgegeben von E. Zeidler*, volume 3. Springer, 2012. doi: 10.1007/978-3-8348-2359-5.
- [164] Johannes Zeitler and Meinard Müller. Reformulating soft dynamic time warping: Insights into target artifacts and prediction quality. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 127–133, Daejeon, South Korea, 2025. doi: 10.5281/ZENODO.17706349.
- [165] Johannes Zeitler and Meinard Müller. A unified perspective on CTC and SDTW using differentiable DTW. *IEEE Transactions on Audio, Speech and Language Processing*, 34:936–951, 2026. doi: 10.1109/TASLP.2026.3657213.
- [166] Johannes Zeitler and Meinard Müller. Subsequence SDTW: Differentiable alignment with flexible boundary conditions. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 15252–15256, Barcelona, Spain, 2026. doi: 10.1109/ICASSP55912.2026.11463857.
- [167] Johannes Zeitler, Simon Deniffel, Michael Krause, and Meinard Müller. Stabilizing training with soft dynamic time warping: A case study for pitch class estimation with weakly aligned targets. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–439, Milano, Italy, 2023. doi: 10.5281/ZENODO.10265317.

- [168] Johannes Zeitler, Michael Krause, and Meinard Müller. Soft dynamic time warping with variable step weights. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 356–360, Seoul, South Korea, 2024. doi: 10.1109/ICASSP48485.2024.10446578.
- [169] Johannes Zeitler, Ben Maman, and Meinard Müller. Robust and accurate audio synchronization using raw features from transcription models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 120–127, San Francisco, USA, 2024. doi: 10.5281/ZENODO.14877291.
- [170] Johannes Zeitler, Christof Weiß, Vlora Arifi-Müller, and Meinard Müller. BPSD: A coherent multi-version dataset for analyzing the first movements of beethoven’s piano sonatas. *Transactions of the International Society for Music Information Retrieval (TISMIR)*, 7(1):195–212, 2024. doi: 10.5334/TISMIR.196.
- [171] Albert Zeyer, Ralf Schlüter, and Hermann Ney. Why does CTC result in peaky behavior? *CoRR*, abs/2105.14849, 2021.
- [172] Huanzhou Zhu, Zhuoer Gu, Haiming Zhao, Keyang Chen, Chang-Tsun Li, and Ligang He. Developing a pattern discovery method in time series data and its GPU acceleration. *Big Data Mining and Analytics*, 1(4): 266–283, 2018. doi: 10.26599/BDMA.2018.9020021.