A Unified Perspective on CTC and Soft-DTW Using Differentiable DTW

Johannes Zeitler, *Student Member, IEEE*, and Meinard Müller, *Fellow, IEEE*

***Abstract*—Training deep neural networks on unaligned sequence data is fundamental to tasks such as automatic speech recognition, lyrics alignment, and music transcription. Strongly aligned annotations, which provide frame-level correspondences between input and target sequences, are often costly, impractical, or unreliable. In contrast, weakly aligned annotations, which specify only segment-level alignment, are more scalable and easier to obtain, but present challenges for training and supervision. A widely used technique for handling weakly aligned data is Connectionist Temporal Classification (CTC). While CTC enables end-to-end training without explicit alignments, it is difficult to interpret, structurally rigid, and relies on a special blank symbol to handle label repetitions. The main contribution of this work is to explore the relationship between CTC and the less commonly used but conceptually simpler Soft Dynamic Time Warping (SDTW), which offers a more intuitive and flexible approach to weak alignment. We introduce a generalization of SDTW that incorporates cell-wise step weights, variable step sizes, and flexible boundary conditions. We refer to this extended framework as Differentiable Dynamic Time Warping (dDTW), which naturally subsumes CTC and SDTW as special cases and provides a unified perspective on these alignment-based losses. We systematically compare SDTW, CTC, and related variants in two controlled and illustrative tasks from music information retrieval, analyzing prediction accuracy, training stability, alignment behavior, and the implications of the blank symbol, in both single- and multi-label problems.**

***Index Terms*—CTC, DTW, SDTW, dDTW, differentiable alignment, music synchronization.**

I. Introduction and Related Work

Training deep neural networks (DNNs) on large amounts of sequential data is a core strategy in various domains, including automatic speech recognition (ASR) [1], motion alignment [2], and music transcription [3]. Ideally, such tasks rely on training data comprising precisely aligned input–target pairs with frame-wise correspondence.

While strongly aligned data are often difficult to obtain, weakly aligned input–target segments, where only the start and end points of regions are known, can be more readily obtained. In ASR, weak targets often consist of the corresponding text or phoneme sequences without explicit timing. In music information retrieval (MIR), such weak targets can be derived for a given music recording from a corresponding score that includes unaligned note content. One approach for training DNNs on weakly aligned pairs of input data and labels is an iterative alignment and prediction refinement in an expectation-maximization (EM)-like fashion [4], [5]. Other examples of model architectures and training paradigms handling weakly aligned data are the attention mechanism [6], optimal transport [7], and the utilization of sequential models like RNNs [8] (see Fig. 1 for a schematic overview of alignment paradigms).

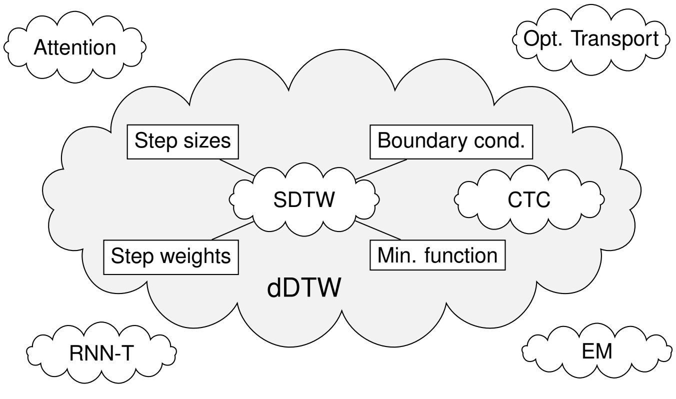

Fig. 1. Schematic illustration of alignment-based loss functions. The dDTW framework generalizes the CTC and SDTW algorithms.

A widely adopted method for end-to-end DNN training with weak supervision is the Connectionist Temporal Classification (CTC) loss [9], 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 image-based sequence recognition [10] and speech recognition [11]. In MIR, CTC has been successfully applied to tasks such as score–audio retrieval [12] and lyrics alignment [13]. To address multi-label scenarios, the classical CTC algorithm has been extended to multi-label CTC (MCTC) [14], and successfully applied to multi-pitch and pitch class estimation tasks [15], [16]. 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 [9], [12], [17]. Furthermore, CTC is inherently limited to feature-to-label tasks with targets from a finite alphabet.

Similarly, Soft Dynamic Time Warping (SDTW) [18],

Manuscript submitted 01 August 2025, revised 15 December 2025 and 08 January 2026.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant No. 500643750 (MU 2686/15-1) and Grant No. 521420645 (MU 2686/17-1).

The authors are with the International Audio Laboratories Erlangen (a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS), Am Wolfsmantel 33, 91058 Erlangen, Germany (e-mail: johannes.zeitler@audiolabs-erlangen.de; meinard.mueller@audiolabs-erlangen.de).

[19] offers a differentiable generalization of classical Dynamic Time Warping (DTW) [20]. By replacing the hard minimum operator with a smooth approximation, SDTW remains conceptually simple and interpretable. Applications of SDTW in MIR include performance–score synchronization [21] and multi-pitch estimation (MPE) [22]. Several extensions have been proposed to improve SDTW, including alternative smoothing strategies [23], improvements regarding mathematical properties [24], training stabilization techniques [25], and customizable step weights [26]. Unlike CTC, SDTW operates on arbitrary cost matrices and is applicable to diverse problem settings, including feature-to-label tasks via one-hot and multi-hot encodings, as well as feature-to-feature alignment tasks using continuous-valued feature representations [27], [28].

In this paper, we establish a formal connection between CTC and SDTW by reformulating the CTC objective to match a generalized SDTW formulation. To reveal a mathematical connection between the two losses, we extend SDTW with three building blocks: parameterizable cell-wise step weights, individualized step sizes, and flexible boundary conditions (see Fig. 1). We refer to this generalized alignment algorithm as Differentiable Dynamic Time Warping (dDTW) and derive efficient dynamic programming (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.

The remainder of this paper is structured as follows. In Sections II and III, we briefly review the objectives of CTC and SDTW, respectively, focusing on their global formulations rather than algorithmic specifics. In Section IV, 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 V 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 VI and experimentally evaluate in Section VII performance of CTC, SDTW, and combinations thereof in the unified dDTW framework. Finally, we summarize our findings and outline future research directions in Section VIII.

II. CTC Revisited

In this section, we describe the CTC objective function as proposed by Graves et al. [9]. 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 [9] for implementation details regarding, e.g., the forward and backward passes using DP.

Notation

Throughout this paper, 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 $[\alpha, \beta] := \{x \in \mathbb{R} | \alpha \leq x \leq \beta\}$; discrete integer ranges by $[1{:}N] := \{1, 2, \ldots, N\}$. Vectors are represented by lowercase letters (e.g., x), and matrices are denoted by uppercase letters (e.g., $\mathbf{X}$), with coefficients given by $\mathbf{X}(n, m)$.

A. Alphabet, Labels, and Predictions

Following the original CTC paper [9], we define an **alphabet** $\mathcal{A} = \{a_1, \ldots, a_I\}$ of available symbols, and extended alphabet $\mathcal{A}' = \mathcal{A} \cup \{\epsilon\}$ including a special blank symbol ϵ. Let $Y = (y_1, \ldots, y_M)$ denote a **label sequence** with labels $y_m \in \mathcal{A}$. 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 [9] by adding the blank symbol before and after every element in Y, resulting in the **extended label sequence** $Y^{\mathrm{e}} = (\epsilon, y_1, \epsilon, \ldots, \epsilon, y_M, \epsilon) = (y^{\mathrm{e}}_1, \ldots, y^{\mathrm{e}}_{M^{\mathrm{e}}})$ with $M^{\mathrm{e}} = 2M + 1$ and $y^{\mathrm{e}}_m \in \mathcal{A}'$. Furthermore, we define output features, also called **prediction sequence** $X = (x_1, \ldots, x_N)$ with $x_n \in \mathcal{F}_X$, where $\mathcal{F}_X$ denotes an abstract feature space, e.g., symbol probabilities.

B. Label Paths

Our goal is to align the label sequence to the prediction sequence. Following [9], we establish notation for alignments in terms of **label paths** $\pi = (\pi_1, \ldots, \pi_N)$, which model the alignment as a sequence of label symbols from the extended alphabet with $\pi_n \in \mathcal{A}'$. To define a validity condition for a label path, a projection operator $\kappa_N : \mathcal{A}'^N \to \mathcal{A}^{\leq N}$ is defined that takes a label path π of length N and first collapses repeated symbols and then removes all blank symbols, yielding a label sequence of length $\leq N$. 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**: π_1 is either the blank symbol ϵ or the first label y_1; in other words, $\pi_1 \in \{y^{\mathrm{e}}_1, y^{\mathrm{e}}_2\}$.
- **End condition**: π_N is either the blank symbol ϵ or the last label y_M; in other words, $\pi_N \in \{y^{\mathrm{e}}_{M^{\mathrm{e}}-1}, y^{\mathrm{e}}_{M^{\mathrm{e}}}\}$.
- **Step sizes**: π is a (weakly) monotonic unfolding of Y^{e}. Leaving out blank symbols is allowed if the preceding

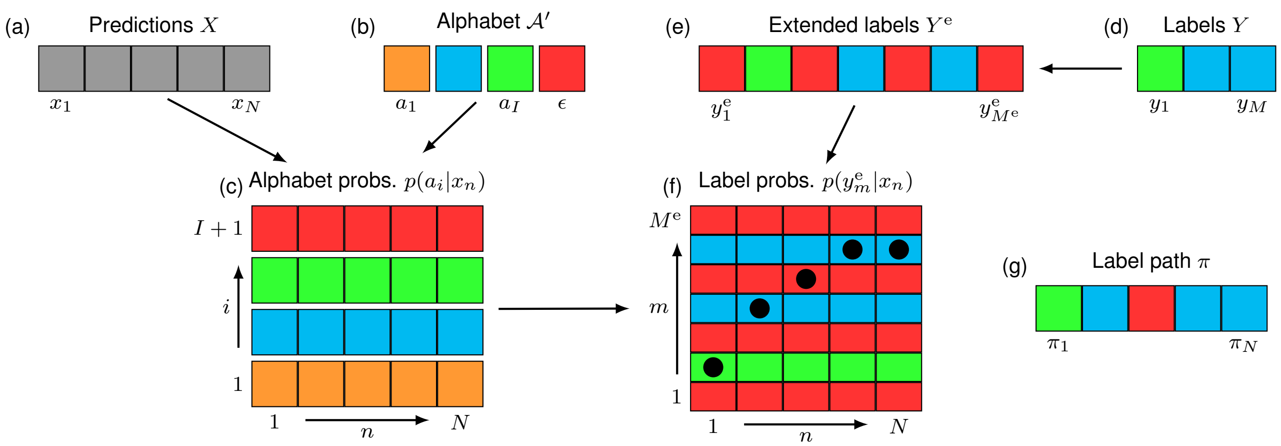

Fig. 2. Overview of CTC loss computation. (a) Prediction sequence X. (b) Extended alphabet $\mathcal{A}'$. (c) Probabilities of alphabet symbols $p(a_i|x_n)$. (d) Label sequence Y. (e) Extended label sequence Y^e. (f) Probabilities of extended labels $p(y^\mathrm{e}_m|x_n)$. (g) Possible label path π corresponding to circles in (f).

and succeeding labels are different. In other words, let $\pi_n = y^\mathrm{e}_m$, then

$$\begin{aligned} &\pi_{n+1} \in \left\{y^\mathrm{e}_m, y^\mathrm{e}_{m+1}, y^\mathrm{e}_{m+2}\right\}, && \text{if } y^\mathrm{e}_m \neq y^\mathrm{e}_{m+2}, \\ &\pi_{n+1} \in \left\{y^\mathrm{e}_m, y^\mathrm{e}_{m+1}\right\}, && \text{else}. \end{aligned} \tag{1}$$

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

C. CTC Objective

Following [9] 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

$$p(\pi|X) = \prod_{n=1}^{N} p(\pi_n|x_n), \tag{2}$$

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 VI. 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:

$$p(Y|X) = \sum_{\pi \in \kappa_N^{-1}(Y)} p(\pi|X). \tag{3}$$

While the direct evaluation of (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 $\mathcal{O}(M^\mathrm{e} \cdot N)$ [9].

D. Practical Implementation

In standard CTC implementations, the process starts by computing probabilities for all symbols in the extended alphabet $\mathcal{A}'$ conditioned on the network predictions X (Fig. 2a-c). The label sequence (Fig. 2d) is then expanded by inserting blanks before and after each label (Fig. 2e), and rows from the alphabet probability matrix are arranged accordingly (Fig. 2f). The loss is computed on this re-arranged matrix by accumulating the probabilities of all valid label paths π (Fig. 2g). For large alphabets (e.g., in multi-pitch estimation), computing probabilities for all symbols becomes computationally expensive (Fig. 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.

III. SDTW Revisited

In this section, we briefly describe the SDTW loss function as proposed by Cuturi et al. [18] by defining the cost matrix as an input to the algorithm, specifying alignment paths, and outlining the global objective of SDTW.

A. Labels, Predictions, and Cost Matrix

The computation of the SDTW loss is based on a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ (Fig. 3c) that is computed from two sequences $X = (x_1, \dots, x_N)$ (Fig. 3a) and $Y = (y_1, \dots, y_M)$ (Fig. 3b) [18] 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

$$\mathbf{C}\,(n, m) = c(\mathbf{x}_n, \mathbf{y}_m), \tag{4}$$

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

B. Alignment Paths

While the label path introduced for CTC is defined w.r.t. the 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

$$\mathcal{I} := [1 : N] \times [1 : M] \tag{5}$$

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 [20]. Valid alignment paths follow a set of constraints, in particular boundary and step size conditions. We specify boundary conditions $\mathcal{B}_\mathrm{start}$ and $\mathcal{B}_\mathrm{end}$ 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., $\mathcal{B}_\mathrm{start}, \mathcal{B}_\mathrm{end} \subseteq \mathcal{I}$. For any

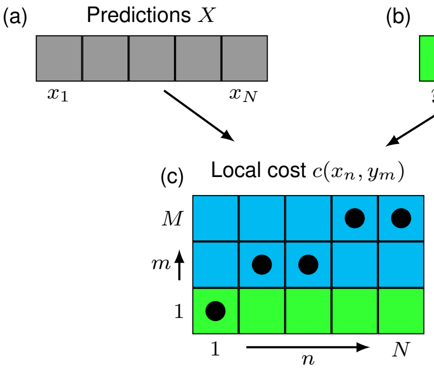

Fig. 3. Overview of SDTW loss computation. (a) Prediction sequence X. (b) Label sequence Y. (c) Cost matrix $\mathbf{C}$. A possible alignment path is illustrated with circles.

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 $\mathcal{S} = \{(i_s, j_s)\,|s \in [1:S]\}$ 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 [18]:

- **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 [20].

C. 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 [18], [19]:

$$\begin{aligned}\mathrm{SDTW}(\mathbf{C}) &= \mu\Big(\Big\{\sum_{(n,m)\in P} \mathbf{C}\,(n, m) \;\Big|\; P \in \mathcal{P}\Big\}\Big) \qquad (6)\\ &= -\gamma \log\Big(\sum_{P\in\mathcal{P}} \exp\Big(\sum_{(n,m)\in P} -\mathbf{C}\,(n, m)\,/\gamma\Big)\Big)\,,\end{aligned}$$

where μ is a differentiable approximation of the minimum function [18], [29]. In particular, we use the softmin function defined as

$$\mu(Q) = -\gamma \log\Big(\sum_{q\in Q} \exp\left(-q/\gamma\right)\Big)\,, \tag{7}$$

for a finite set $Q \subset \mathbb{R}$, where γ is a temperature parameter controlling the smoothness of the approximation. It can be shown that the SDTW cost can be computed efficiently by a DP recursion as long as μ is defined as in (7) [19].

IV. Relating CTC and SDTW

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

A. 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 [9]. Starting from (2) and (3), we obtain $\mathcal{L}_{\text{CTC}}$, which we then bring to a log-sum-exp formulation by introducing redundant exponential and log terms:

$$\begin{aligned}\mathcal{L}_{\text{CTC}}\,(X, Y) &= -\log p\,(Y|X) \qquad (8)\\ &= -\log\Big(\sum_{\pi\in\kappa_N^{-1}(Y)} \prod_{n=1}^{N} p\,(\pi_n|x_n)\Big)\\ &= -\log\Big(\sum_{\pi\in\kappa_N^{-1}(Y)} \prod_{n=1}^{N} \exp\left(\log p\,(\pi_n|x_n)\right)\Big)\\ &= -\log\Big(\sum_{\pi\in\kappa_N^{-1}(Y)} \exp\Big(\sum_{n=1}^{N} \log p\,(\pi_n|x_n)\Big)\Big)\,.\end{aligned}$$

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

$$\mathbf{C}(n, m) = -\log p(y^{\text{e}}_m|x_n)\,. \tag{9}$$

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}:

$$\begin{aligned}\mathcal{L}_{\text{CTC}}\,(X, Y) &= -\log \sum_{P\in\mathcal{P}} \exp\left(\sum_{(n,m)\in P} \log p\,(y^{\text{e}}_m|x_n)\right)\\ &= -\gamma \log \sum_{P\in\mathcal{P}} \exp\left(\sum_{(n,m)\in P} -\mathbf{C}\,(n, m)\,/\gamma\right)\\ &= \mathrm{SDTW}\,(\mathbf{C})\;, \qquad (10)\end{aligned}$$

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.

B. Necessary Properties for Equivalence of CTC and SDTW

To ensure mathematical equivalence in (10), all alignment paths P must fulfill certain properties, as illustrated in Fig. 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 $\mathcal{B}_{\text{end}} = \{(N, M^{\text{e}} - 1), (N, M^{\text{e}})\}$. 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.

C. Conclusion

While the SDTW formulation from (6) can be evaluated for arbitrary alignment paths P including the modified boundary and step size conditions presented in Section IV-B, the exhaustive search over all possible alignment paths $\mathcal{P}$ is computationally inefficient. To address this, efficient DP recursions have been proposed [18], which yield the optimal result of the global formulation (6) when using the softmin approximation from (7) [19]. However, the original SDTW formulation from Cuturi and Blondel [18] 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 [19] propose a differentiable DP framework for general weighted directed acyclic graphs, encompassing algorithms such as Viterbi [30], SDTW [18], and attention mechanisms [6]. 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.

V. Differentiable Dynamic Time Warping

In the following, we introduce Differentiable Dynamic Time Warping (dDTW), an extension of SDTW [18] that supports flexible boundary conditions, arbitrary step sizes [19] and step weights (extending [26]), and general minimum functions [19]. 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:

$$(n, m) \prec (n', m') \quad \text{if} \quad n \leq n',\ m \leq m',\ (n, m) \neq (n', m')\,. \tag{11}$$

A. dDTW Building Blocks

Our proposed dDTW algorithm is based on efficient forward and backward DP recursions, building upon [18]. We define the dDTW algorithm with four building blocks, namely the boundary conditions, the allowed step sizes, the step weights associated to 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 Fig. 4.

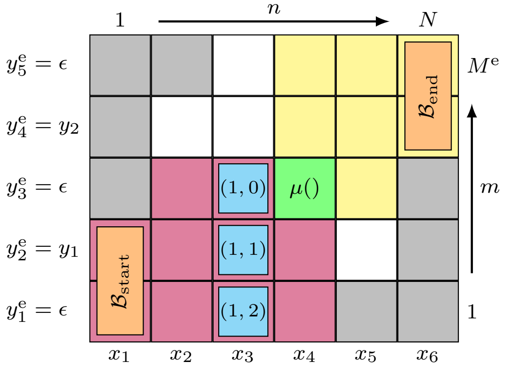
1
n
N
$y_5^{\text{e}} = \epsilon$
$y_4^{\text{e}} = y_2$
$y_3^{\text{e}} = \epsilon$
$y_2^{\text{e}} = y_1$
$y_1^{\text{e}} = \epsilon$
M^{e}
m
1
$\mathcal{B}_{\text{end}}$
$\mathcal{B}_{\text{start}}$
$(1, 0)$
$\mu()$
$(1, 1)$
$(1, 2)$
x_1
x_2
x_3
x_4
x_5
x_6

Fig. 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.

1) 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 the 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., $\mathcal{B}_{\text{end}} = \{(N, M - 1), (N, M)\}$.

2) Step Sizes: We define the allowed step sizes as a set of integer tuples. In the case of 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.

3) Step Weights: While individual weights associated with each step size were proposed in prior work [26], 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.

4) Minimum Function: We denote by μ a differentiable approximation of the minimum function. In this work, we exclusively use the softmin function as given in (7), with the gradient $\nabla\mu : \mathbb{R}^S \to \mathbb{R}^S$ calculated as

$$[\nabla\mu(Q)]_s = \frac{\exp\left(-q_s/\gamma\right)}{\sum_{q_t \in Q} \exp\left(-q_t/\gamma\right)} \tag{12}$$

for an ordered set of real numbers $Q = \{q_1, \ldots, q_S\}$. Note that dDTW can be used with arbitrary differentiable mini-

Algorithm 1 DP forward pass for dDTW

```
Inputs:
  Cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ indexed by cells $\mathcal{I}$
  Boundary conditions $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$
  Step sizes $\mathcal{S} = \{(i_s, j_s) \mid s \in [1:S]\}$
  Step weights $\mathbf{W} \in \mathbb{R}_+^{N \times M \times S}$
  Minimum function $\mu : \mathbb{R}^S \to \mathbb{R}$
for $m = 1, ..., M$ do
  for $n = 1, ..., N$ do
    for $s = 1, ..., S$ do
      $\mathbf{f}_s^{(n,m)} = \begin{cases} \infty, & \text{if } (n-i_s, m-j_s) \notin \mathcal{I}\,, \\ \mathbf{D}(n{-}i_s, m{-}j_s) + \mathbf{W}(n,m,s)\mathbf{C}(n,m), & \text{else}\,, \end{cases}$
    end for
    $\mathbf{f}_{S+1}^{(n,m)} = \begin{cases} \infty, & \text{if } (n,m) \notin \mathcal{B}_{\text{start}}\,, \\ \mathbf{C}(n,m), & \text{else}\,. \end{cases}$
    $\mathbf{D}(n,m) = \mu\left(\mathbf{f}^{(n,m)}\right)$
    Save $\mathbf{B}(n,m) = \nabla\mu\left(\mathbf{f}^{(n,m)}\right)$ for backward
  end for
end for
$\mathbf{b} = \{\mathbf{D}(n,m) \mid (n,m) \in \mathcal{B}_{\text{end}}\}$
$\text{dDTW}(\mathbf{C}) = \mu(\mathbf{b})$
Output: dDTW cost $\text{dDTW}(\mathbf{C}) \in \mathbb{R}$
```

Algorithm 2 DP backward pass for dDTW

```
Inputs:
  Backtracking tensor $\mathbf{B} \in \mathbb{R}^{N \times M \times S}$
  Boundary conditions $\mathcal{B}_{\text{start}}, \mathcal{B}_{\text{end}} \subseteq \mathcal{I}$
  End boundary costs $\mathbf{b} := \left\{\mathbf{b}^{(n,m)} \in \mathbb{R} \mid (n,m) \in \mathcal{B}_{\text{end}}\right\}$
  Step weights $\mathbf{W} \in \mathbb{R}_+^{N \times M \times S}$
  Step sizes $\mathcal{S} = \{(i_s, j_s) \mid s \in [1:S]\}$
for $m = M, ..., 1$ do
  for $n = N, ..., 1$ do
    for $s = 1, \ldots, S$ do
      $\frac{\partial\, \text{dDTW}(\mathbf{C})}{\partial\, \mathbf{D}(n+i_s, m+j_s)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{E}(n+i_s, m+j_s), & \text{else}\,. \end{cases}$
      $\frac{\partial\, \mathbf{D}(n+i_s, m+j_s)}{\partial\, \mathbf{D}(n,m)} = \begin{cases} 0, & \text{if } (n+i_s, m+j_s) \notin \mathcal{I}, \\ \mathbf{B}(n+i_s, m+j_s, s), & \text{else}\,. \end{cases}$
    end for
    $[\nabla\mu(\mathbf{b})]^{(n,m)} = \begin{cases} \partial\, \mu(\mathbf{b}) / \partial\, \mathbf{b}^{(n,m)}, & \text{if } (n,m) \in \mathcal{B}_{\text{end}}, \\ 0, & \text{else}\,. \end{cases}$
    $\mathbf{E}(n,m) = \sum_{s=1}^{S} \frac{\partial\, \text{dDTW}(\mathbf{C})}{\partial\, \mathbf{D}(n+i_s, m+j_s)} \cdot \frac{\partial\, \mathbf{D}(n+i_s, m+j_s)}{\partial\, \mathbf{D}(n,m)} + [\nabla\mu(\mathbf{b})]^{(n,m)}$
    for $s = 1, ..., S+1$ do
      $\frac{\partial\, \mathbf{f}_s^{(n,m)}}{\partial\, \mathbf{C}(n,m)} = \begin{cases} \mathbf{W}(n,m,s), & \text{if } s \in [1:S], \\ 1, & \text{if } s{=}S{+}1 \text{ and } (n,m) \in \mathcal{B}_{\text{start}}, \\ 0, & \text{else}\,. \end{cases}$
    end for
    $\mathbf{G}(n,m) = \sum_{s=1}^{S+1} \mathbf{B}(n,m,s) \cdot \frac{\partial\, \mathbf{f}_s^{(n,m)}}{\partial\, \mathbf{C}(n,m)}$
    $\mathbf{H}(n,m) = \mathbf{E}(n,m) \cdot \mathbf{G}(n,m)$
  end for
end for
Output: Gradient $\mathbf{H} \in \mathbb{R}^{N \times M}$
```

mum functions, such as the "sparsemin" function [19], [31]. However, in this work, we concentrate on the soft minimum function. 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.

B. Forward Recursion

The dDTW algorithm takes as input a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$, calculated as described in (4). Following [18], 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 $(i_s, j_s) \in \mathcal{S}$:

$$\mathbf{f}_s^{(n,m)} := \begin{cases} \infty, & \text{if } (n-i_s, m-j_s) \notin \mathcal{I}\,, \\ \mathbf{D}(n{-}i_s, m{-}j_s) + \mathbf{W}(n,m,s)\mathbf{C}(n,m), & \text{else}\,, \end{cases} \tag{13}$$

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

$$\mathbf{f}_{S+1}^{(n,m)} := \begin{cases} \infty, & \text{if } (n,m) \notin \mathcal{B}_{\text{start}}\,, \\ \mathbf{C}(n,m), & \text{else}\,. \end{cases} \tag{14}$$

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

$$\mathbf{D}(n,m) = \mu\left(\mathbf{f}^{(n,m)}\right)\,. \tag{15}$$

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

$$\mathbf{B}(n,m) := \nabla\mu\left(\mathbf{f}^{(n,m)}\right) \in \mathbb{R}^{S+1}\,, \tag{16}$$

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 $\mathbf{b} := \left\{\mathbf{b}^{(n,m)} \in \mathbb{R} \mid (n,m) \in \mathcal{B}_{\text{end}}\right\}$ with elements $\mathbf{b}^{(n,m)} = \mathbf{D}(n,m)$. 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 $\mathbf{b}$:

$$\text{dDTW}(\mathbf{C}) := \mu(\mathbf{b})\,. \tag{17}$$

The complete forward algorithm for dDTW is summarized in Algorithm 1.

C. Backward Recursion

Our goal is to obtain the gradient of the total cost $\text{dDTW}(\mathbf{C})$ 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 [18]. This concept is similar to the backtracking of classical DTW [20].

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 $\mathbf{E}(n,m)$) [18]:

$$\begin{aligned}\mathbf{H}(n,m) &:= \frac{\partial\,\mathrm{dDTW}\,(\mathbf{C})}{\partial\,\mathbf{C}(n,m)}\\ &= \frac{\partial\,\mathrm{dDTW}\,(\mathbf{C})}{\partial\,\mathbf{D}(n,m)}\cdot\frac{\partial\,\mathbf{D}(n,m)}{\partial\,\mathbf{C}(n,m)}\\ &= \mathbf{E}(n,m)\cdot\mathbf{G}(n,m)\,,\end{aligned}\tag{18}$$

where we define $\mathbf{E}(n,m) := \partial\,\mathrm{dDTW}(\mathbf{D})/\partial\,\mathbf{D}(n,m)$ 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:

$$\begin{aligned}\mathbf{E}(n,m) = \sum_{s=1}^{S}&\frac{\partial\,\mathrm{dDTW}\,(\mathbf{C})}{\partial\,\mathbf{D}(n+i_s,m+j_s)}\cdot\frac{\partial\,\mathbf{D}(n+i_s,m+j_s)}{\partial\,\mathbf{D}(n,m)}\\ &+[\nabla\mu(\mathbf{b})]^{(n,m)}\,,\end{aligned}\tag{19}$$

with $(i_s,j_s)\in\mathcal{S}$. The first term in (19) denotes the influence of a following cell on the total cost and, as in standard SDTW [18], is given by previously computed entries of the $\mathbf{E}$-matrix:

$$\frac{\partial\,\mathrm{dDTW}\,(\mathbf{C})}{\partial\,\mathbf{D}(n+i_s,m+j_s)} = \begin{cases}0, & \text{if } (n+i_s,m+j_s)\notin\mathcal{I},\\ \mathbf{E}(n+i_s,m+j_s), & \text{else}\,.\end{cases}\tag{20}$$

The second term in (19), also found in standard SDTW [18], denotes the influence of the current cell on a following cell and is given by the previously computed backtracking matrix:

$$\frac{\partial\,\mathbf{D}(n+i_s,m+j_s)}{\partial\,\mathbf{D}(n,m)} = \begin{cases}0, & \text{if } (n+i_s,m+j_s)\notin\mathcal{I},\\ \mathbf{B}(n+i_s,m+j_s,s), & \text{else}\,.\end{cases}\tag{21}$$

The third term in (19) denotes potential contributions coming from the end boundary conditions:

$$[\nabla\mu(\mathbf{b})]^{(n,m)} = \begin{cases}\partial\,\mu(\mathbf{b})/\partial\,\mathbf{b}^{(n,m)}, & \text{if } (n,m)\in\mathcal{B}_{\mathrm{end}},\\ 0, & \text{else}\,.\end{cases}\tag{22}$$

In other words, (19) describes the influence of a cell's accumulated cost $\mathbf{D}(n,m)$ on the total alignment cost $\mathrm{dDTW}\,(\mathbf{C})$ 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 $\mathrm{dDTW}\,(\mathbf{C})$, 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 [20]. By again applying the chain rule, we obtain $\mathbf{G}(n,m)$ as

$$\mathbf{G}(n,m) = \sum_{s=1}^{S+1}\frac{\partial\,\mathbf{D}(n,m)}{\partial\,\mathbf{f}_s^{(n,m)}}\cdot\frac{\partial\,\mathbf{f}_s^{(n,m)}}{\partial\,\mathbf{C}(n,m)}\,,\tag{23}$$

where the first term in (23) is given by the backtracking matrix

$$\frac{\partial\,\mathbf{D}(n,m)}{\partial\,\mathbf{f}_s^{(n,m)}} = \mathbf{B}(n,m,s)\,,\tag{24}$$

and the second term in (23) is given by either the step weight, or, in the case (n,m) is in the start boundary conditions, a factor of one:

$$\frac{\partial\,\mathbf{f}_s^{(n,m)}}{\partial\,\mathbf{C}(n,m)} = \begin{cases}\mathbf{W}(n,m,s), & \text{if } s\in[1:S],\\ 1, & \text{if } s{=}S{+}1 \text{ and } (n,m)\in\mathcal{B}_{\mathrm{start}},\\ 0, & \text{else}\,.\end{cases}\tag{25}$$

The full backward pass is summarized in Algorithm 2. If we use unweighted dDTW, where all step weights are one ($\mathbf{W}$ is an all-ones matrix), $\mathbf{G}$ also becomes an all-ones matrix and thus $\mathbf{H}=\mathbf{E}$ [26]. The matrix $\mathbf{E}$ can be interpreted as a "soft alignment matrix" [18], 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 Fig. 10 and refer to [18], [25], [26] for further explanations.

VI. 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 $p(y_m^{\mathrm{e}}|x_n)$ 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 (main-melody pitch class prediction, see Section VII-B) and the other being a multi-label scenario (polyphonic pitch class estimation from music recordings, see Section VII-C). Both tasks, which are well-studied in previous work [12], [16], [25], [26], serve three purposes: First, they illustrate how to define suitable label probabilities $p(y_m^{\mathrm{e}}|x_n)$ 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.

A. 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 the main melody (i.e., the musical theme, see Fig. 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 [12]. The extended alphabet $\mathcal{A}'$ 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., $y_m^{\mathrm{e}}\in\{0,1\}^D$ (see Fig. 6). $D\in\{12,13\}$ is chosen depending on the requirement of an additional blank class. For example, in the score from Fig. 5, we model the first note of the annotated theme (pitch class "D") as a one-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.

1) 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.

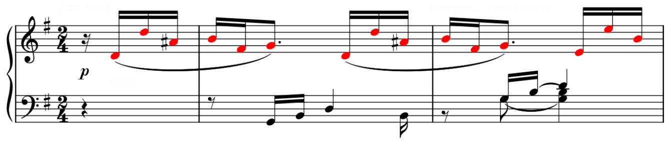

Fig. 5. Musical score of the first measures of Beethoven's piano sonata Op. 14 No. 2, first movement. The main melody (theme) is highlighted in red.

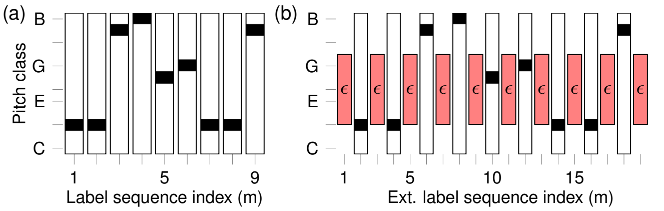

Fig. 6. Single-class target representations for the pitch classes corresponding to the main melody (theme) of the running example in Fig. 5 for the first two measures. (a) Label sequence Y. (b) Extended label sequence Y^{e}.

Using vector notation, the label probabilities can be obtained as

$$p(y^{\mathrm{e}}_m|x_n) = \langle x_n, y^{\mathrm{e}}_m\rangle\,, \tag{26}$$

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^{e}_m.

2) SDTW: For SDTW, we do not require the blank symbol (i.e., $D = 12$) and directly use the weak labels $Y = (y_1, \ldots, y_M)$ as the target sequence. To mirror the CTC label probability specified in (26) in SDTW, we define the cost function as the negative log-likelihood for the true class label

$$c(x_n, y_m) = -\log\langle x_n, y_m\rangle \tag{27}$$

and, while disregarding the blank symbol, fulfill the requirement $c(x_n, y_m) = -\log p(y^{\mathrm{e}}_m|x_n)$ given by (10). To avoid changing the network architecture, we model rests as all-zero vectors.

B. 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 multimodal Schubert Winterreise dataset (SWD) [32]. The musical score (see Fig. 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 [33] or as features for music synchronization [34]. We represent the multi-class labels $Y = (y_1, \ldots, y_M)$ 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 $\{C, E^\flat, G\}$ and is encoded as a multi-hot vector $y_1 = [1,0,0,1,0,0,0,1,0,0,0,0]$ (see Fig. 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 $D \in \{12, 13\}$).

1) Multi-Label CTC: In the original CTC formulation, all possible labels must be drawn from an alphabet $\mathcal{A}$. For a multi-label task with 12 pitch classes, this implies $|\mathcal{A}| = 2^{12} = 4096$ possible label combinations. Computing probabilities for all such combinations is infeasible in practice. Therefore, prior work on the multi-label CTC (MCTC) [14], [16] restricts the alphabet to only those combinations actually occurring in the label sequence Y. In our setup, we directly define the conditional probabilities $p(y^{\mathrm{e}}_m|x_n)$ for the elements of the extended label sequence y^{e}_m (see Fig. 8b), which can be used to compute the CTC loss as in (10). Two cases must be distinguished based on the index m: If m is odd, $y^{\mathrm{e}}_m = \epsilon$ (blank symbol), and if m is even, $y^{\mathrm{e}}_m = y_{m/2}$ (the corresponding multi-hot label from Y), as defined in Section II-A. Following [14], [16], we define multi-class probabilities

$$p(y^{\mathrm{e}}_m|x_n) = \begin{cases} p(\epsilon|x_n), & \text{if } m \text{ is odd (blank)},\\ (p(\bar{\epsilon}|x_n)\cdot p(y_{m/2}|x_n), & \text{else (not blank)}\,, \end{cases} \tag{28}$$

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 $y_{m/2} = y^{\mathrm{e}}_m$. 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:

$$p(\epsilon|x_n) = x_{n,0}, \quad p(\bar{\epsilon}|x_n) = 1 - x_{n,0}. \tag{29}$$

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:

$$p(y_m|x_n) = \prod_{i=1}^{12}\Big(y_{m,i}x_{n,i} + (1-y_{m,i})(1-x_{n,i})\Big)\,, \tag{30}$$

assuming conditional independence of pitch classes.

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

$$c_{\mathrm{BCE}}(x_n, y_m) = -\sum_{i=1}^{12}\Big(y_{m,i}\log x_{n,i} + (1-y_{m,i})\log(1-x_{n,i})\Big), \tag{31}$$

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

C. Discussion

In both the single- and the multi-label configurations above, we observe equality in the cost values $c_{\mathrm{BCE}}(x_n, y_m) = -\log p(y_m|x_n)$, aligning with the requirement formulated in (10). Although the CTC and SDTW objectives appear similar in form, they differ conceptually: First, CTC requires explicit modeling of a blank symbol,

This article has been accepted for publication in IEEE Transactions on Audio, Speech and Language Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASLPRO.2026.3657213

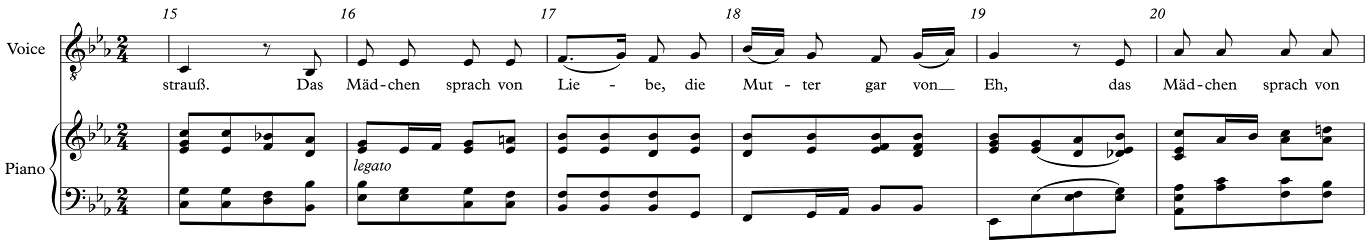
15
16
17
18
19
20
Voice
Piano
strauß. Das Mäd-chen sprach von Lie - be, die Mut - ter gar von__ Eh, das Mäd-chen sprach von
legato

Fig. 7. Excerpt from "Gute Nacht" (Schubert's "Winterreise"), used as a running example for multi-label classification.

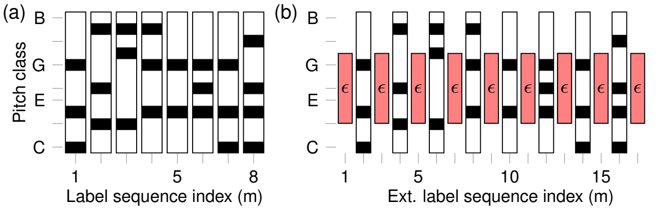

Fig. 8. Multi-class target representation for the pitch classes of the running example in Fig. 7 for measures 15 and 16. (a) Label sequence Y. (b) Extended label sequence Y^{e}.

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.

VII. 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 VII-A), 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, softmin 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, main-melody pitch class prediction (Section VII-B), and evaluate frame-level accuracy and downstream database retrieval given a melody query. 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 VII-C) in a polyphonic pitch-class estimation problem using both MCTC and SDTW, illustrating how SDTW extends to multi-label scenarios. Finally, in Section VII-D we compare the computational cost in practice of classical SDTW and CTC implementations with our proposed dDTW framework. Throughout all experiments, we remove all temporal information from the targets (see Figs. 6 and 8), which may lead to substantial sequence-length mismatches between predictions and labels. Although up-/downsampling strategies can stabilize DTW and SDTW alignments in such cases [35], [22], we do not adopt them here to preserve comparability with CTC and reduce computational cost.

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, we will analyze in detail whether the blank is essential for successful sequence decoding, the separation of repeated symbols, and alignment stabilization, or whether the same effects can instead be achieved through an appropriately parameterized SDTW variant. Code to reproduce all experiments is available at `github.com/groupmm/dDTW_CTC`.

A. Training Configurations

We now discuss the parameterization of various loss functions within our proposed dDTW framework. Table I provides an overview of the dDTW parameterizations leading to different training configurations. In Table I, 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.

1) CTC: For all `CTC` configurations, we adopt the parameterization from Section VI. 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 $c(x_n, y^{\mathrm{e}}_m)$ as defined in (26) and (28). 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 $\mathcal{B}_{\mathrm{start}} = \{(1,1),(1,2)\}$ and $\mathcal{B}_{\mathrm{end}} = \{(N, M^{\mathrm{e}}-1),(N, M^{\mathrm{e}})\}$. 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

	has	(1,0)		(1,1)		(1,2)		(0,1)
ID	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

TABLE I
OVERVIEW OF DIFFERENT MCTC AND SDTW CONFIGURATIONS WITHIN THE DDTW FRAMEWORK. FOR EXPLANATIONS, SEE SECTION VII-A.

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 $+Z \cdot 217$

TABLE II
CNN ARCHITECTURE FOR MAIN-MELODY PITCH CLASS PREDICTION FROM [12]. $Z \in \{0, 1\}$ DENOTES AN OPTIONAL OUTPUT DIMENSION FOR BLANK.

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 Fig. 4.

2) 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 VI, the local cost function is given by (27) and (31). 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 sizes $\mathcal{S} = \{(1, 0), (1, 1)\}$ 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.

3) 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 [26]. Specifically, reducing this weight improves alignment quality. To systematically investigate this, we run all experiments with either the standard horizontal step weight of $w_{\text{h}} = 1.0$ or a reduced weight of $w_{\text{h}} = 0.1$ (suffix -W).

4) Baselines: As a baseline, we use an expectation-maximization (EM)-like approach inspired by [4], [5], 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).

B. Single-Label Classification: Main Melody Prediction

As a first scenario, we consider a single-label classification task: predicting the pitch-class sequence of the main melody (i.e., the musical theme) in audio recordings [12]. We use the musical theme dataset (MTD) [36], 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 [12], we train a small CNN to produce a chromagram-like representation that highlights the pitch classes of the main melody (the musical theme). Each training example pairs an audio segment containing a musical theme with the corresponding pitch-class sequence extracted from the score (see Figs. 5 and 6).

The setup is closely related to ASR training, where utterances are paired with unaligned label sequences; accordingly, [12] optimizes the network using the CTC loss. However, unlike ASR, no transcription is decoded but the posterior probabilities are used 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 theme annotations).

1) Experimental Setup: Throughout all experiments, we follow the training setup of [12] and vary only the loss function within our dDTW framework. The convolutional neural network (CNN) takes as input a harmonic constant-Q transform (HCQT) [37] 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 II and consists of 2D convolutional layers with leaky ReLU activations and a partly trainable pooling stage that reduces the output to D classes (see [12] for details). We train with mini-batches of eight using Adam [38] 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.

2) 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 subsequence DTW [20]. Rankings follow

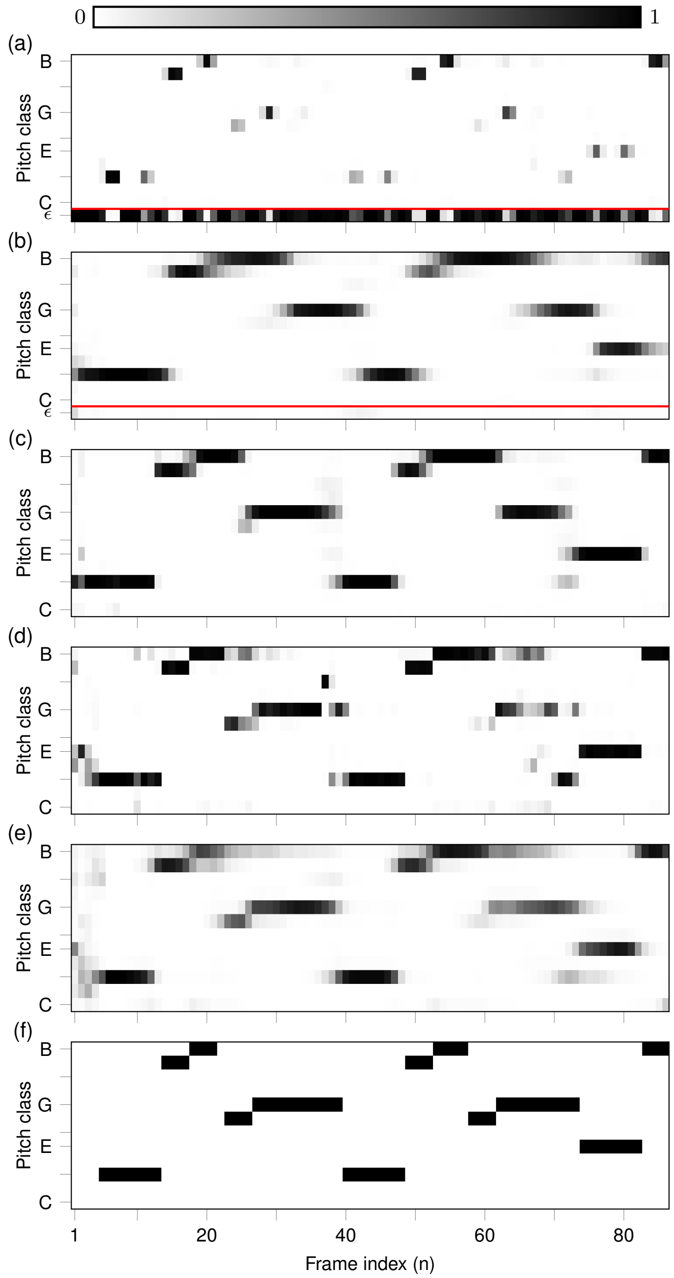

Fig. 9. Predicted pitch classes for the musical theme visualized in Fig. 5 for different network configurations. (a) `CTC-A`. (b) `CTC-C`. (c) `SDTW-E`. (d) `EM`. (e) `strong`. (f) Reference annotations.

ID	Accuracy ↑		Retrieval rank ↑			MRR ↑
	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

TABLE III
EXPERIMENTAL RESULTS FOR MONOPHONIC PITCH-CLASS ENHANCEMENT BASED ON MTD [36]. 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.

the minimum subsequence DTW distance per query–document pair; we report top-k retrieval rates and mean reciprocal rank (MRR) [12]. Results are summarized in Table III with example predictions in Fig. 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 (Fig. 9a), yet achieves strong retrieval (top-1 $= 0.870$). Penalizing blanks (`CTC-B`) or forbidding blank alignments (`CTC-C`) increases accuracy to 0.555 and 0.588 and removes blank predictions (Fig. 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 (Fig. 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 frame-wise accuracy (0.676), but shows temporal blurring (Fig. 9e), which likely explains slightly weaker retrieval than the best SDTW models.

3) 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 [8] 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 [39] between decoded and reference sequences, reported as label error rate (LER), analogous to ASR character error rate.

With horizontal step weight $w_{\mathrm{h}} = 1.0$, 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 (Fig. 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 Fig. 9a), whereas SDTW and strong targets emphasize frame-wise activations rather than onsets (Fig. 9c,e). Addressing this would require explicit onset modeling [40], which is beyond our scope.

To allow a fair comparison between CTC- and SDTW-based models in the single-label setting, we also apply beam-search decoding, a common post-processing strategy in sequence

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

TABLE IV

EXPERIMENTAL RESULTS FOR SYMBOL-LEVEL DECODING ON MTD [36]. 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.

modeling [8]. As shown in Table IV, the trend is consistent: for all configurations except CTC-A, beam search increases LER, sometimes sharply, especially for reduced w_h, where the LER can approach or exceed 1.0. In contrast, only CTC-A, the setting for which beam search was introduced [8], benefits from it, reducing mean LER to 0.256 and achieving the best-case LER of 0.189. These results suggest that the CTC blank symbol functions as a robust "time-filler" in uncertain regions, causing many beam paths to collapse to the same output sequence and thus stabilizing decoding. In contrast, the absence of such a mechanism in SDTW leads to spurious symbol insertions and over-segmentation of the hypothesis space, resulting in unstable sequence estimates.

C. 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 [33]. Unlike larger music transcription models that require pretraining on strongly aligned data [40], this setup enables training from scratch using only weakly aligned data. To ensure comparability with prior work, we use the SWD dataset [32] and a musically motivated CNN architecture from [33]. This setup has previously been explored with strongly aligned targets [33], and with weakly aligned targets using MCTC [16] or SDTW [25], [26].

1) Experimental Setup: The CNN receives as input a HCQT [37], computed from audio at a sampling rate of $22\,050\,\mathrm{Hz}$ with a hop size of 512 samples, resulting in a frame rate of approximately $43\,\mathrm{Hz}$. The HCQT spans six octaves with three bins per semitone (resulting in 216 pitch bins) and includes six harmonic channels (one subharmonic 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 V. We refer to [33], [16] for the design motivation and further details.

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			$48315 + 73 \cdot Z$

TABLE V

CNN ARCHITECTURE FOR PITCH CLASS ESTIMATION FROM [16], [33]. $Z \in \{0,1\}$ DENOTES AN OPTIONAL OUTPUT DIMENSION FOR BLANK.

The SWD contains approximately $11\,\mathrm{h}$ 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 [38] 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.

For qualitative illustration, we use a $10\,\mathrm{s}$ excerpt from the song "Gute Nacht" performed by Randall Scarlata, with the corresponding score shown in Fig. 7 and strongly aligned reference targets shown in Fig. 10f.

2) 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, Fig. 11 reports the minimum, median, and maximum F-measure over five training runs. For selected, didactically relevant configurations, Fig. 10 visualizes the predictions X and soft alignment matrix $\mathbf{E}$ of the model with the lowest validation loss, applied to the running example from Fig. 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 $w_\mathrm{h} = 1.0$, corresponding to the standard configurations in both CTC and SDTW.

Starting with the original CTC algorithm (CTC-A), the test F-measure shown in Fig. 11 is relatively low at 0.81, and shows high variance across different runs. The predictions in Fig. 10a 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.

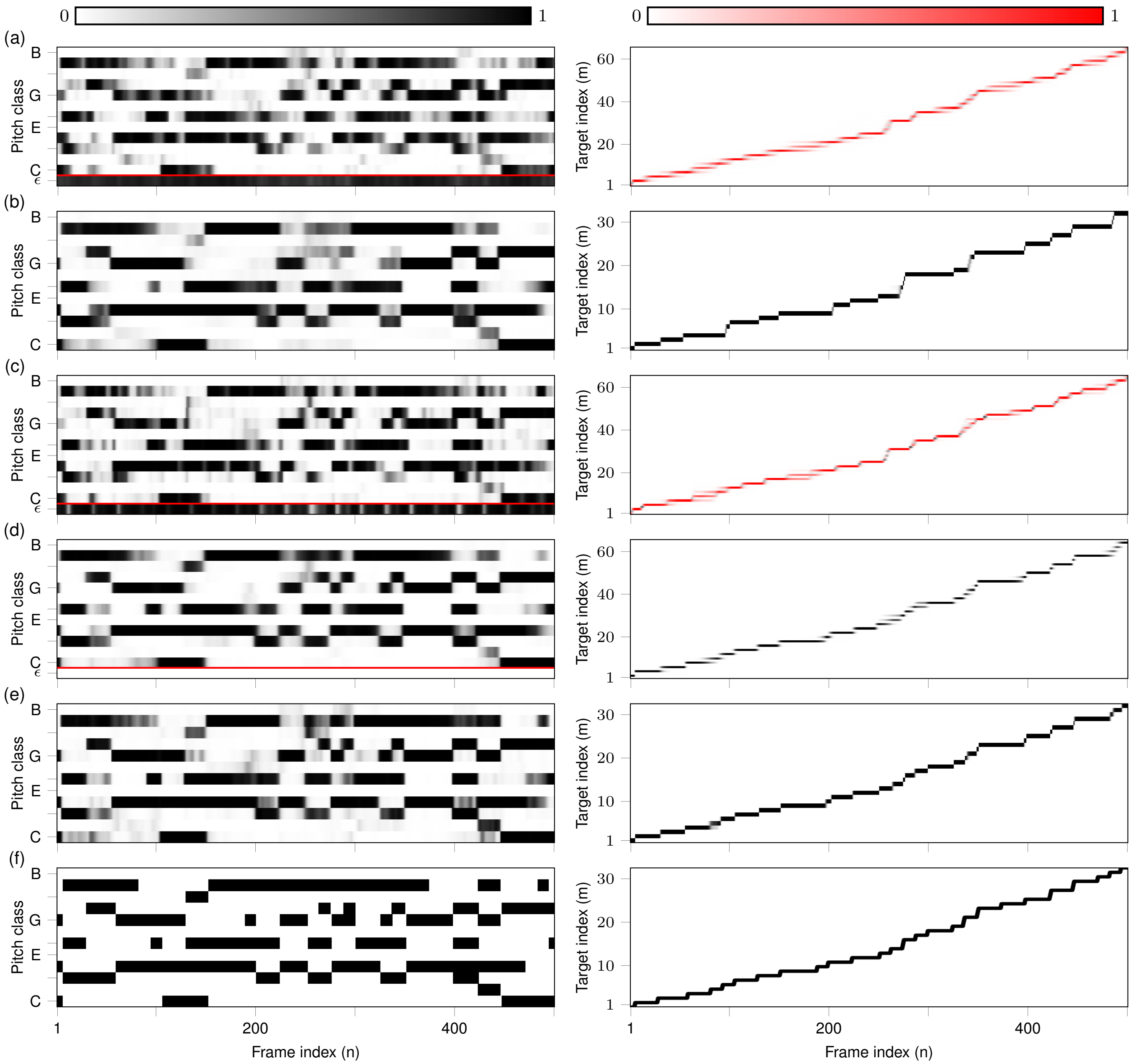

Fig. 10. Predictions (left) and soft alignments (right) for different network configurations for the running example. (a) CTC-A. (b) SDTW-E. (c) CTC-A-W. (d) CTC-B-W. (e) SDTW-E-W. (f) Strongly aligned labels and alignment obtained from reference annotations available in the SWD. Red color is used for the blank symbol ϵ.

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.

Next, we impose the largest possible penalty by setting an infinite weight for steps in and to the blank symbol (CTC-C). According to Fig. 11, 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 Fig. 11 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 Fig. 10b show temporally stable but slightly blurred outputs. The soft alignment matrix reveals some alignment instabilities compared to the reference alignment in Fig. 10f, with certain targets being aligned over extended durations, while others are

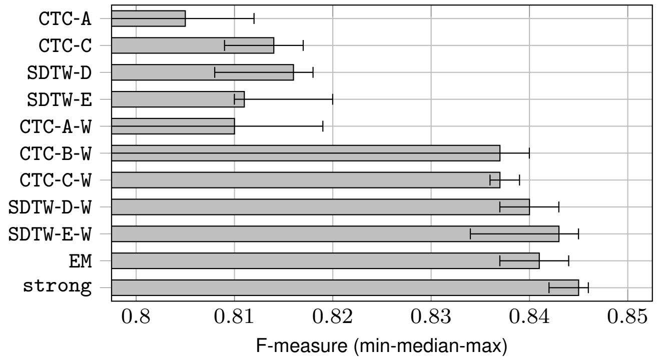

Fig. 11. Test results for the five training runs of each model configuration. The bar represents the median F-measure for frame activation over the five training runs, the whiskers represent the minimum and maximum F-measure of the training runs.

assigned to only short intervals. We address this issue next by lowering the horizontal step weight to enforce more stable target repetitions.

3) Ablation: Sensitivity w.r.t. Horizontal Step Weight: Prior work on SDTW shows that lowering the horizontal step weight w_{h} stabilizes training and improves alignments [26]. 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 by setting the weight $w_{\mathrm{h}} = 0.1$ for the horizontal $(1, 0)$ step.

For the original CTC (CTC-A-W), Fig. 11a shows an improvement in median and maximum F-measure versus CTC-A, though some runs do not converge to a good solution. Predictions in Fig. 10c reveal slightly reduced probabilities for the blank symbol compared to CTC-A (Fig. 10a), 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-measure considerably (Fig. 11), 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 Fig. 10d 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 Fig. 11 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 Fig. 10e show sharp, temporally stable outputs with minimal fluctuations, and the soft alignment matrix resembles the reference alignment to a high degree.

	SDTW		dDTW		CTC	
(N, M)	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

TABLE VI
OVERVIEW OF RUNTIME (MILLISECONDS) AND PEAK MEMORY CONSUMPTION (MEGABYTES) FOR THE SDTW, DDTW, AND CTC LOSS FUNCTIONS FOR DIFFERENT SEQUENCE LENGTHS (N, M).

D. Runtime and Memory Consumption

We analyze runtime and memory usage of three alignment-loss implementations: standard SDTW[1] [41], our dDTW toolbox[2], and the optimized C++ PyTorch CTC loss[3] [42]. 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 $\mathbf{Y}_{\mathrm{CTC}} \in [1 : D]^{B\times M}$. 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 $\mathcal{O}(NM)$ recursions but can be parallelized along anti-diagonals [43], [41], yielding $\mathcal{O}(N + M)$ parallel time. The strictly monotonic CTC dynamic program runs in $\mathcal{O}(N)$ parallel steps. All methods parallelize over the batch, though only dDTW and CTC support variable-length sequences within a batch.

Table VI 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.

E. 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 within a single framework, enabling fine-grained control of loss parameters and providing a continuous interpolation between the two conceptually

[1] https://github.com/Maghoumi/pytorch-softdtw-cuda
[2] https://github.com/groupmm/dDTW_CTC
[3] https://github.com/pytorch/pytorch

different losses. In the following, we highlight three key differences observed in our experiments.

1) 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.

2) 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 $\mathcal{O}(NM)$ implementation, this would double runtime. In practice, GPU-parallel implementations of SDTW [41] 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 [42], which outperform research-oriented SDTW and dDTW implementations by a wide margin.

3) 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 [40], [44].

VIII. Conclusion

In this paper, 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 parameterizations that 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 modeling repeated symbols in dDTW without relying on a blank symbol by, e.g., using onset models, the usage of dDTW as a subsequence alignment method, and task-specific choices of the differentiable minimum function. Additionally, while this work used dDTW only to compare CTC and SDTW, we plan to investigate the generalization of other alignment paradigms via dDTW.

IX. Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant No. 500643750 (MU 2686/15-1) and Grant No. 521420645 (MU 2686/17-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS.

References

[1] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, Beijing, China, 2014, pp. 1764–1772.

[2] C. Chang, D. Huang, Y. Sui, L. Fei-Fei, and J. C. Niebles, "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)*, Long Beach, CA, USA, 2019, pp. 3546–3555.

[3] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[4] B. Maman and A. H. Bermano, "Unaligned supervision for automatic music transcription in the wild," in *Proceedings of the International Conference on Machine Learning (ICML)*, Baltimore, Maryland, USA, 2022, pp. 14 918–14 934.

[5] X. Riley, D. Edwards, and S. Dixon, "High resolution guitar transcription via domain adaptation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seoul, South Korea, 2024, pp. 1051–1055.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 5998–6008.

[7] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, vol. 2, 2013, pp. 2292—-2300.

[8] A. Graves, "Sequence transduction with recurrent neural networks," 2012. [Online]. Available: https://arxiv.org/abs/1211.3711

[9] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, Pittsburgh, Pennsylvania, USA, 2006, pp. 369–376.

[10] B. Shi, X. Bai, and C. 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*, vol. 39, no. 11, pp. 2298–2304, 2017. [Online]. Available: https://doi.org/10.1109/TPAMI.2016.2646371

[11] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html

[12] F. Zalkow and M. Müller, "CTC-based learning of chroma features for score-audio music retrieval," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2957–2971, 2021.

[13] D. Stoller, S. Durand, and S. 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)*, Brighton, UK, 2019, pp. 181–185.

[14] C. Wigington, B. L. Price, and S. Cohen, "Multi-label connectionist temporal classification," in *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, 2019, pp. 979–986.

[15] C. Weiß and G. 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)*, New Paltz, USA, 2021, pp. 121–125.

[16] ——, "Training deep pitch-class representations with a multi-label CTC loss," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021, pp. 754–761.

[17] A. Zeyer, R. Schlüter, and H. Ney, "Why does CTC result in peaky behavior?" *CoRR*, vol. abs/2105.14849, 2021. [Online]. Available: https://arxiv.org/abs/2105.14849

[18] M. Cuturi and M. Blondel, "Soft-DTW: a differentiable loss function for time-series," in *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia, 2017, pp. 894–903. [Online]. Available: http://proceedings.mlr.press/v70/cuturi17a.html

[19] A. Mensch and M. Blondel, "Differentiable dynamic programming for structured prediction and attention," in *Proceedings of the International Conference on Machine Learning (ICML)*, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 3459–3468. [Online]. Available: http://proceedings.mlr.press/v80/mensch18a.html

[20] M. Müller, *Fundamentals of Music Processing – Using Python and Jupyter Notebooks*, 2nd ed. Springer Verlag, 2021.

[21] R. Agrawal, D. Wolff, and S. Dixon, "A convolutional-attentional neural framework for structure-aware performance-score synchronization," *IEEE Signal Processing Letters*, vol. 29, pp. 344–348, 2021.

[22] M. Krause, C. Weiß, and M. 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)*, Rhodes Island, Greece, 2023, pp. 1–5.

[23] I. Hadji, K. G. Derpanis, and A. D. Jepson, "Representation learning via global temporal alignment and cycle-consistency," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Virtual, 2021, pp. 11 068–11 077.

[24] M. Blondel, A. Mensch, and J. Vert, "Differentiable divergences between time series," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Virtual, 2021, pp. 3853–3861. [Online]. Available: http://proceedings.mlr.press/v130/blondel21a.html

[25] J. Zeitler, S. Deniffel, M. Krause, and M. 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)*, Milano, Italy, 2023, pp. 433–439.

[26] J. Zeitler, M. Krause, and M. Müller, "Soft dynamic time warping with variable step weights," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seoul, South Korea, 2024, pp. 356–360.

[27] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 1801–1810.

[28] M. Krause, S. Strahl, and M. 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.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[30] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[31] M. Blondel and V. Roulet, "The elements of differentiable programming," 2025. [Online]. Available: https://arxiv.org/abs/2403.14606

[32] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. Grohganz, "Schubert Winterreise dataset: A multimodal scenario for music analysis," *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 14, no. 2, pp. 25:1–18, 2021.

[33] C. Weiß, J. Zeitler, T. Zunner, F. Schuberth, and M. Müller, "Learning pitch-class representations from score–audio pairs of classical music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021, pp. 746–753.

[34] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.

[35] J. Kraprayoon, A. Pham, and T. J. Tsai, "Improving the robustness of DTW to global time warping conditions in audio synchronization," *Applied Sciences*, vol. 14, no. 4, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/4/1459

[36] F. Zalkow, S. Balke, V. Arifi-Müller, and M. Müller, "MTD: A multimodal dataset of musical themes for MIR research," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 180–192, 2020.

[37] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 tracking in polyphonic music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 63–70.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, California, USA, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[39] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[40] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the International Society for Music Information Retrieval Conference, (ISMIR)*, Paris, France, 2018, pp. 50–57.

[41] M. Maghoumi, E. M. Taranta, and J. LaViola, "DeepNAG: Deep non-adversarial gesture generation," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, College Station, Texas, USA, 2021, pp. 213–223.

[42] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[43] H. Zhu, Z. Gu, H. Zhao, K. Chen, C. Li, and L. He, "Developing a pattern discovery method in time series data and its GPU acceleration," *Big Data Mining and Analytics*, vol. 1, no. 4, pp. 266–283, 2018. [Online]. Available: https://doi.org/10.26599/BDMA.2018.9020021

[44] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE/ACM Transactions of Audio, Speech, and Language Processing*, vol. 29, pp. 3707–3717, 2021. [Online]. Available: https://doi.org/10.1109/TASLP.2021.3121991

Johannes Zeitler (johannes.zeitler@audiolabs-erlangen.de) received the B.Sc. degree in electrical engineering and the M.Sc. degree in signal processing and communications engineering from Friedrich-Alexander Universität Erlangen-Nürnberg, Germany, in 2019 and 2021, respectively. He joined the International Audio Laboratories Erlangen, Germany, in 2022, where he is currently pursuing his Ph.D. under the supervision of Prof. Meinard Müller. His research interests include alignment techniques in music processing. In 2025, he co-presented the tutorial "Differentiable Alignment Techniques for Music Processing" at the International Society for Music Information Retrieval (ISMIR) Conference in Daejeon, South Korea.

Meinard Müller (meinard.mueller@audiolabs-erlangen.de) received the Diploma degree (1997) in mathematics and the Ph.D. degree (2001) in computer science from the University of Bonn, Germany. After his postdoctoral studies (2001-2003) in Japan and his habilitation (2003-2007) in multimedia retrieval in Bonn, he worked as a senior researcher at Saarland University and the Max-Planck Institut für Informatik (2007-2012). Since 2012, he has held a professorship for Semantic Audio Signal Processing at the International Audio Laboratories Erlangen, a joint institute of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer Institute for Integrated Circuits IIS. His recent research interests include music processing, music information retrieval, audio signal processing, and motion processing. He was a member of the IEEE Audio and Acoustic Signal Processing Technical Committee (2010-2015), a member of the Senior Editorial Board of the IEEE Signal Processing Magazine (2018-2022), and a member of the Board of Directors, International Society for Music Information Retrieval (2009-2021, being its president in 2020/2021). In 2020, he was elevated to IEEE Fellow for contributions to music signal processing.