Friedrich-Alexander-Universität Erlangen-Nürnberg

Internship Report

Training with Unaligned Dataset
Soft Dynamic Time Warping

submitted by

Quang Hoang Nguyen Vo

submitted

October 1, 2025

Supervisors

M.Sc. Johannes Zeitler
Prof. Dr. Meinard Müller

International Audio Laboratories Erlangen
A joint institution of the
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and
the Fraunhofer-Institut für Integrierte Schaltungen IIS.

Abstract

The evolution of Deep Neural Networks has shifted the paradigm of music information retrieval from heuristic and mathematical models to data-driven approaches, which rely on large amounts of labelled training data. However, it introduces challenges when training with weakly aligned datasets. In this project, we investigate the characteristics of differential dynamic time warping through soft dynamic time warping (SDTW) algorithm when training with weakly aligned data. The main objective is to integrate SDTW as a loss function in the training process of a chord recognition model. The dataset will have its chord label timestamps distorted or removed to simulate weakly or unaligned data. The SDTW loss function will then be used to train the model with the distorted dataset. Our results show that SDTW is able to achieve comparable performance to the baseline model trained on strongly aligned data with a slight decrease in F1-score.

Contents

Chapter 1

Introduction

Amidst the rapid advancement of deep neural networks (DNNs), The field of music information retrieval (MIR) has witnessed a significant shift from traditional heuristic and mathematical models to data-driven approaches that heavily rely on large amounts of labelled training data. such as pitch estimation [7], audio embeddings [2], automatic music transcription [1].

However, the reliance on large and accurate datasets poses many challenges, considering the time-consuming and labor-intensive nature of manual annotation, as well as the potential for human error and subjectivity. Thus, it is generally difficult to obtain strongly aligned annotations, where each frame of the audio signal is associated with a corresponding label. Instead, weakly aligned or unaligned annotations are more common, where only the presence or absence of certain labels is known, without precise temporal alignment. This eases the data acquisition process, but requires a more sophisticated loss function to train the model. One proposed solution is using connectionist temporal classification (CTC) loss [5]. Another proposed approach is to use soft dynamic time warping (SDTW) [3], which is what we will explore in this project.

Dynamic Time Warping (DTW) is a well-known algorithm for measuring similarity between two temporal sequences that may vary in speed or timing [10]. One limitation of DTW is its non-differentiability due to the hard minimum operator used in the alignment cost computation. Therefore, it cannot be used directly as a loss function in DNNs as it requires the computation of gradients during backpropagation [11, 4]. SDTW addresses this limitation by smoothing the minimum operator, making it differentiable at all points [3]. This opens up the possibility of using SDTW as a loss function in DNNs, allowing the model to learn from weakly aligned or unaligned data.

In this project, we investigate the characteristics of SDTW and implement it as a loss function in the training process of DNNs. We choose the task of chord recognition as a case study [9]. Furthermore, we compare the results of the model trained with SDTW loss against a baseline model trained with binary cross-entropy (BCE) loss on strongly aligned data, and analyze the gradient computation during training. The structure of the report is as follows: Chapter 2 presents the formulation of SDTW. Then we describe the experimental setup in Chapter 3, followed by the evaluation of the results in Chapter 4. Finally, we conclude the report in Chapter 5. Starting from the next chapter, we will refer weakly aligned data as soft targets, and strongly aligned data as strong targets.

Chapter 2

Soft Dynamic Time Warping Algorithm

In this chapter, we present the mathematical formulation of the SDTW algorithm. We closely follow the paper by Zeitler et al. [12] for consistency and clarity in notation and terminology.

2.1 Definition and Notation

Let $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_M)$ and $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$, where $\mathbf{X}$ and $\mathbf{Y}$ representing the predicted and soft target sequences, respectively. Furthermore, we assume $M > N$. The objective of SDTW is to calculate the alignment cost between $\mathbf{X}$ and $\mathbf{Y}$.

2.2 Forward Pass

In the forward pass, we compute the accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{M \times N}$, where each element $D(i, j)$ represents the minimum cost of aligning the first i elements of $\mathbf{X}$ with the first j elements of $\mathbf{Y}$. The accumulated cost is computed using the local cost matrix $\mathbf{C} \in \mathbb{R}^{M \times N}$, where $\mathbf{C}(i, j) = c(\mathbf{x}_i, \mathbf{y}_j)$, which measures the dissimilarity between the elements $\mathbf{x}_i$ and $\mathbf{y}_j$. A common choice for the local cost function is the squared Euclidean distance:

$$\mathbf{C}(i, j) = \|\mathbf{x}_i - \mathbf{y}_j\|_2^2. \tag{2.1}$$

Instead of using the hard minimum operator as in traditional DTW, SDTW employs a differentiable approximation, defined as:

$$\min{}^{\gamma}(\mathcal{S}) = -\gamma \log \sum_{s \in \mathcal{S}} \left(e^{-s/\gamma} \right), \tag{2.2}$$

where $\gamma > 0$ is a smoothing parameter that controls the softness of the minimum operation, and $\mathcal{S}$ is the set of values over which the soft minimum is computed. As γ approaches 0, the soft minimum converges to the hard minimum operator. Combining with the step constraint of DTW, we only consider three possible predecessor cells for each cell (i, j), which are $(i-1, j)$, $(i, j-1)$, and $(i-1, j-1)$. Thus the accumulated cost matrix $\mathbf{D}$ is computed recursively as follows:

$$\mathbf{D}(i,j) = \mathbf{C}(i,j) + \min{}^{\gamma}\left(\mathbf{D}(i-1,j), \mathbf{D}(i,j-1), \mathbf{D}(i-1,j-1)\right). \tag{2.3}$$

The final alignment cost is given by

$$\mathrm{SDTW}^{\gamma}_{\mathbf{C}} = \mathbf{D}(M,N). \tag{2.4}$$

The forward pass is summarized in Algorithm 1 in the appendix.

2.3 Backward Pass

To train the model, we need to compute the gradients of the SDTW loss with respect to the model parameters [12]. The backward pass computes the gradient $\mathbf{H} \in \mathbb{R}^{M \times N}$ of the SDTW cost with respect to the cost matrix $\mathbf{C}$. An efficient way to compute the gradient is to use a dynamic programming approach similar to the forward pass. The gradient is computed recursively as follows:

$$\mathbf{H}(i,j) = \frac{\partial \mathbf{D}(M,N)}{\partial \mathbf{D}(i,j)} \frac{\partial \mathbf{D}(i,j)}{\partial \mathbf{C}(i,j)} \tag{2.5}$$

During the backward pass, we consider the predecessor cells of (i, j) in the forward pass. This expands Equation 2.5 to:

$$\mathbf{H}(i,j) = \frac{\partial \mathbf{D}(M,N)}{\partial \mathbf{D}(i+1,j)} \frac{\partial \mathbf{D}(i+1,j)}{\partial \mathbf{D}(i,j)} + \frac{\partial \mathbf{D}(M,N)}{\partial \mathbf{D}(i,j+1)} \frac{\partial \mathbf{D}(i,j+1)}{\partial \mathbf{D}(i,j)} + \frac{\partial \mathbf{D}(M,N)}{\partial \mathbf{D}(i+1,j+1)} \frac{\partial \mathbf{D}(i+1,j+1)}{\partial \mathbf{D}(i,j)}. \tag{2.6}$$

For simplicity, we denote $\mathbf{E}(i,j) = \frac{\partial \mathbf{D}(M,N)}{\partial \mathbf{D}(i,j)}$ and $\mathbf{F}^{p,q}(i,j) = \frac{\partial \mathbf{D}(i+p,j+q)}{\partial \mathbf{D}(i,j)}$, where $p, q \in \{0, 1\}$.

Using the derivative of the soft minimum operator w.r.t. to its input $s \in \mathcal{S}$

$$\frac{\partial \min^{\gamma}(\mathcal{S})}{\partial s} = \frac{e^{-s/\gamma}}{\sum_{s' \in \mathcal{S}} e^{-s'/\gamma}}, \tag{2.7}$$

we can compute $\mathbf{F}^{1,0}(i,j)$ without loss of generality as:

$$\mathbf{F}^{1,0}(i,j) = \frac{\partial \min^{\gamma}(\mathbf{D}(i+1,j))}{\partial \mathbf{D}(i,j)} = \frac{e^{-(\mathbf{C}(i+1,j) - \mathbf{D}(i,j))/\gamma}}{e^{-\mathbf{D}(i,j)/\gamma}}. \tag{2.8}$$

Similarly, we can compute $\mathbf{F}^{0,1}(i,j)$ and $\mathbf{F}^{1,1}(i,j)$. Then we can reformulate $\mathbf{E}$ as:

$$\mathbf{E}(i,j) = \mathbf{E}(i+1,j)\mathbf{F}^{1,0}(i,j) + \mathbf{E}(i,j+1)\mathbf{F}^{0,1}(i,j) + \mathbf{E}(i+1,j+1)\mathbf{F}^{1,1}(i,j). \tag{2.9}$$

Denote $\mathbf{G}(i,j) = \frac{\partial \mathbf{D}(i,j)}{\partial \mathbf{C}(i,j)}$, we have the final formulation of $\mathbf{H}$ as:

$$\mathbf{H}(i,j) = \mathbf{E}(i,j)\mathbf{G}(i,j). \tag{2.10}$$

An overview of the backward pass is provided in Algorithm 2 in the appendix. Note that the recursive computation of both forward and backward pass has a time complexity of $\mathcal{O}(MN)$.

Chapter 3

Experimental Setup

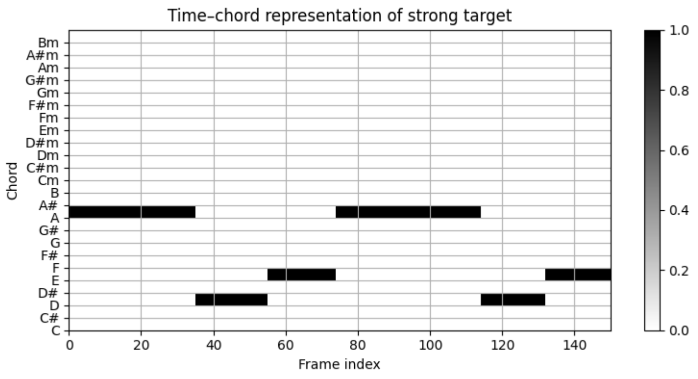
In this section, we present the objectives of our experiment, the dataset used for training, alongside the network architecture and the training process.

3.1 Dataset

For the experiment, we use the Beatles dataset retrieved from Isophonics [6], consisting of four audio recordings with respective annotations. Since the original annotations have more than 24 chord types, which would make the network too complex and beyond the scope of this project, we therefore consider the simplified version of such annotations, which reduces the number of chord types to only 24 (12 root notes with their respective major or minor variant)[9]. We split the dataset into training, validation, and test sets. For the test set, a short segment of "Let It Be" is used, while the rest of the dataset is split into a 3:1 ratio for training and validation.

We chose a sequence length of 150 samples for training and validating the model, creating 43 and 12 segments for training and validation, respectively. In case of soft alignment, we remove the adjacent repetitions in the sequence as shown in Figure 3.1.

However, this method introduces a problem where batching target sequences of different lengths is not possible. To address this issue, after reduction, we pad the sequences with repeating elements until they reach a desired length, or "soft length". We propose two padding strategies: uniform and last element repetition, as illustrated in Figure 3.2. We define uniform padding as repeating all elements in the sequence, i.e "stretching" the sequence to the desired length, whereas the last element only repeats the last element until the desired length is reached. Due to this, the soft target may have a length that is shorter, equal to, or longer than the original strong target. After some experiments, we find that a soft length of 16 covers all of the possible reduced sequences while keeping the reduction ratio high. We also choose a soft length of 75 to evaluate the effect of a different reduction ratio on the performance of the model. Both of the chosen lengths are strictly less than the original length of 150 to be consistent with our assumption that $M > N$ in Chapter 2.



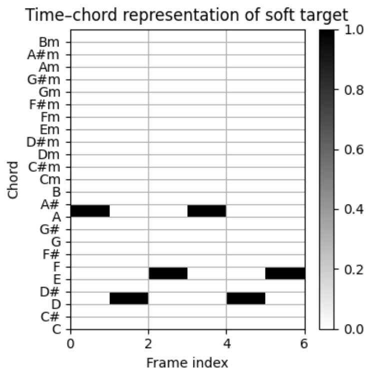


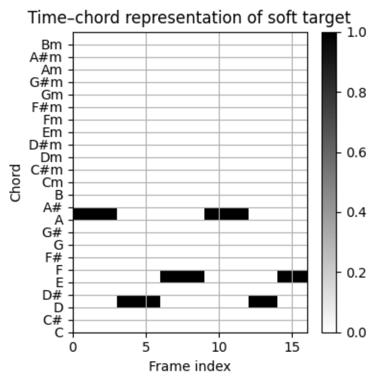
Figure 3.1: Example of strong and soft target sequences. The strong target sequence has a length of 150, while the soft target sequence has a length of 6 after removing adjacent repetitions.

Layer	Input Dimension	Output Dimension	Parameters
Log-compression	(12, 150)	(12, 150)	1
Normalization	(12, 150)	(12, 150)	0
dChord	(12, 150)	(24, 150)	24
softmax	(24, 150)	(24, 150)	0

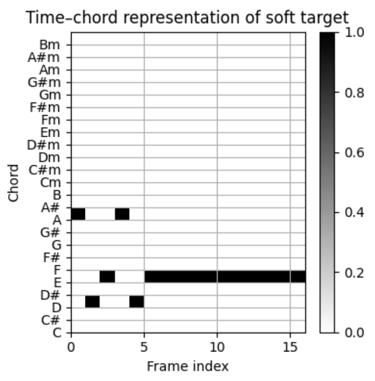
Table 3.1: Architecture of the chord recognition network. Input sequence length is 150 samples.

3.2 Model Architecture

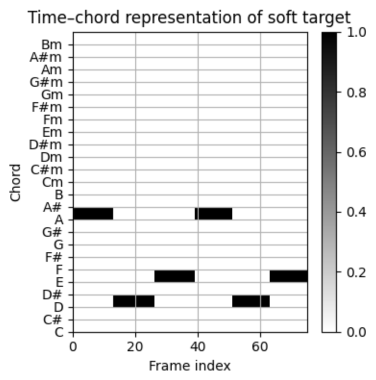
Given that the aim of this experiment is to evaluate the performance of the proposed SDTW loss function, the network architecture plays a minor role and is kept simple. Therefore, we used a simple chord recognition network (dChord) [13] that is based on the template-based chord recognition algorithm [9]. This network consists of a single layer that acts as the chord template to predict a 24-dimensional activation vector, corresponding to 24 chord types. The network has a total of 25 trainable parameters. Table 3.1 illustrates the components of the architecture with their respective input and output dimensions.



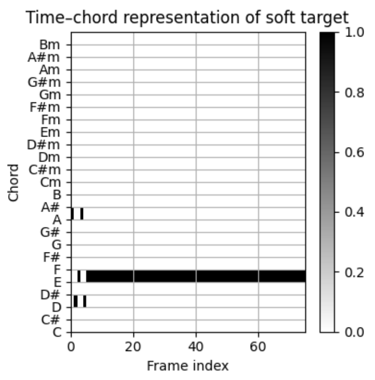
(a) Uniform padding with soft length of 16.



(b) Last element repetition with soft length of 16.



(c) Uniform padding with soft length of 75.



(d) Last element repetition with soft length of 75.

Figure 3.2: Example of different padding strategies with different soft lengths. The original strong target sequence has a length of 150, while the soft target sequence has a length of 5 after removing adjacent repetitions. The padded sequences have lengths of 16 and 75, respectively.

During the training process, we use Adam Optimizer [8] with a a learning rate $\alpha = 0.1$ with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Since the dataset is small, of only 43 training segments, we use a single batch of 43 targets to train the model over 300 epochs for both targets. We also enable weight sharing, where the weights of the chords in the same chord class (major or minor) are shared, so that the model can compensate for the chords that are underrepresented in either chord class. We conduct the training process on Dell Inc. XPS 15 9570 with Intel Core i7-8750G, 16 GB RAM, with Ubuntu 24.04.2 LTS Operating System.

Chapter 4

Evaluation

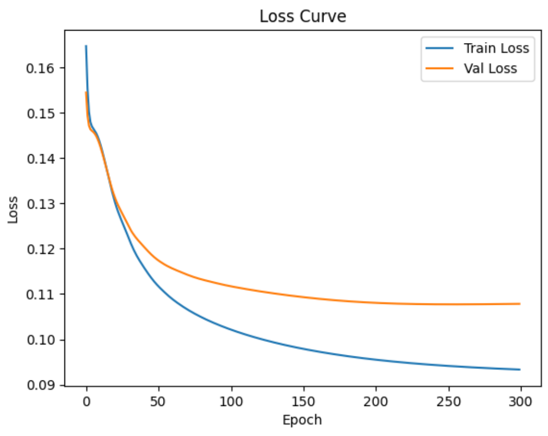
In this section, we evaluate the performance of the aforementioned SDTW loss function against the baseline model trained with BCE loss for the task of chord recognition. We train each model five times to observe any randomness present in the training process, and the model consistently achieves identical performance metrics across multiple runs. We train the models on a total of five cases, including one case of strong targets and four soft cases. The soft cases cover all combinations of soft lengths, which are 16 and 75, and two different padding strategies (uniform and last element repetition).

4.1 Training Curve

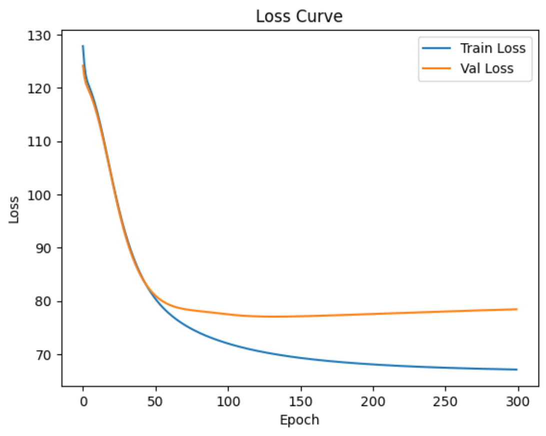
Figure 4.1 shows the training and validation loss curves of the models trained on strong and soft targets. Overall, we observe that the training curves of both models exhibit a similar trend, with the training loss decreasing over time. However, there are some notable differences between the two curves. The loss value for the model trained on the soft targets is much larger than that of the model trained on the strong targets, which ranges from 130 to 70, compared to 0.16 to 0.11 for the model trained on the strong targets. On strong targets, the validation loss converges around epoch 230. On soft targets, the model improves until epoch 130, then the validation loss starts to increase while the training loss continues to decrease. In general, the training curve show that using SDTW loss allows the model to learn from the soft targets, with a training curve similar to that of the model trained with BCE loss.

4.2 Training results

Table 4.1 summarizes the results of the baseline model trained on strong targets with binary cross-entropy loss and the model trained on soft targets with SDTW loss. Here, we choose two different soft lengths, 16 and 75, to evaluate the effect of the reduction ratio on the performance. We also compare two different padding strategies, uniform and last element repetition, making a total of four cases for the soft targets. The performance is evaluated using the F1-score metric on the test set, which is used in template-based chord recognition task [9].



(a) Training and validation loss curves of the model trained on strong targets with BCE loss.



(b) Training and validation loss curves of the model trained on soft targets with SDTW loss.

Figure 4.1: Training and validation loss curves of the models trained on strong and soft targets.

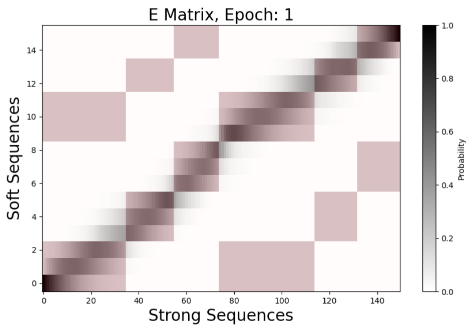
Target/ Original Length	Loss Function (Pad Strategy)	Test F1-score (%)
150/150	BCE	77.5
16/150	SDTW (uniform)	76.1
75/150	SDTW (uniform)	76.1
16/150	SDTW (last)	76.1
75/150	SDTW (last)	68.1

Table 4.1: Results of the models trained on strong and soft targets.

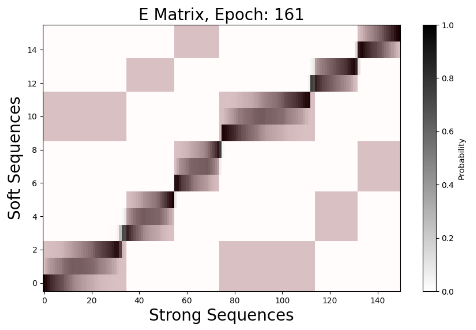
Overall, we see that SDTW loss is able to achieve comparable performance to the baseline model trained on strong targets, with a slight decrease in F1-score of 1.4%. This shows that SDTW can effectively learn from the soft targets that contain less or noisy temporal information, making it more robust to variations in the input sequence. Observing the results of different padding strategies, we find that last element repetition leads to identical performance as uniform padding for short soft lengths, though it eventually degrades as the soft length increases (up to 8% on soft length 75). This is likely due to the repetition, which introduces a bias towards the last element, as demonstrated in Figure 3.2. Nonetheless, the result shows that SDTW is rather flexible, being able to compensate for small mistakes in annotations. Additionally, the weight distribution of the trained model is provided in the Appendix B.

4.3 Gradient Analysis

Inspecting the SDTW gradient during training, with different soft lengths and uniform padding, we can observe the change in probability distribution of the alignment path. Figure 4.2 and 4.3 demonstrate the computed gradient at epoch 1 and epoch 161 of the training process with different soft lengths and padding strategies. We chose epoch 161 as this is when the validation loss starts

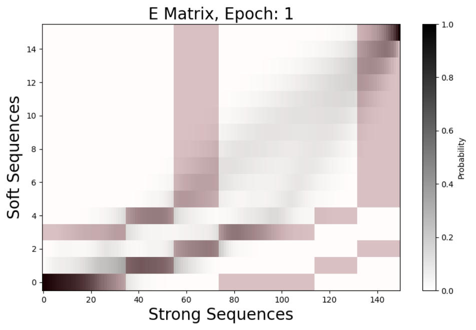


(a) Epoch 1

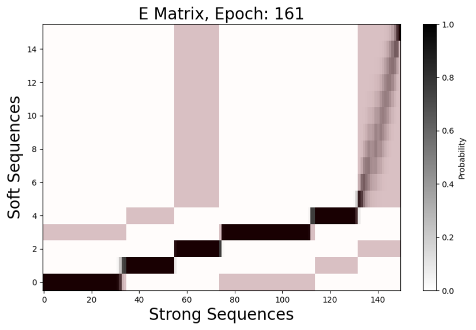


(b) Epoch 161

Figure 4.2: Gradient over epochs with soft length 16 and uniform padding.



(a) Epoch 1



(b) Epoch 161

Figure 4.3: Gradient over epochs with soft length 16 and last element repetition padding.

to increase. The regions, where the chord is the same on both sequences, are highlighted in light red rectangles. Here, the ground truth alignment is the middle diagonal path that connects the top-left and bottom-right corners of the matrix.

We notice that at first the gradient shows a diagonal prior to the alignment path (at epoch 1), indicating that the model is initially uncertain about the alignment. However, as training progresses, the gradient shifts towards the ground truth, albeit not perfectly contained within the ground truth region (at epoch 161). This indicates that the model is capable of learning the alignment in case of soft targets, despite the lack of precise temporal information. Moreover, we inspect the gradient with more extreme alignment paths, such as with last element repetition, shown in Figure 4.3. In such cases, the gradient behaviour becomes more pronounced. It at first creates two distinct paths with similar probabilities, as it quickly converges to the ground truth alignment from the initial diagonal prior. Although these cases may cause a large drop in model performance, as shown in section 4.2, we can see the flexibility of SDTW loss, being able to adapt to different alignment paths.

Despite following the ground truth, the gradient distribution appears to be blurry and not sharply concentrated along the alignment path. This is likely due to the choice of soft temperature γ, which controls the softness of the minimum operator in SDTW. In appendix C, we provide an analysis of the effect of γ on the gradient distribution.

Chapter 5

Conclusions

In this report, we have presented a comprehensive study on the use of the SDTW loss for chord recognition in music information retrieval.

Despite the simplicity of the network architecture, the results indicate that the model is capable of effectively predicting unseen data, even when the training data contains missing or imprecise temporal information, with only a slight degradation in performance. Our experiments provide a method to circumvent the batching limitations when working with variable-length sequences, and we have demonstrated that the model trained with soft targets is capable of generalizing to unseen data, as evidenced by the improved performance metrics on the validation set. The use of soft targets and SDTW loss allows the model to learn from a richer set of annotations, where imperfections regarding the timing and duration of musical events may occurred.

Furthermore, our analysis of the model's behavior reveals interesting insights into the learning process. The gradient during training proves that SDTW can effectively find the time warping alignment between the input and target sequences; therefore, it mitigates the impact of misalignments and improves the overall robustness of the model.

Several limitations of our approach should be acknowledged. First, the small and imbalanced data set used for the experiment may not fully capture the diversity of musical styles and chord progressions, which causes an bias towards major chords. Second is the method of providing soft targets. The current approach starts from strong targets and reduces the repetitions in order to emulate misalignments, but it may not fully capture the nuances of real-world annotations. Finally, the proposed method to solve variable-length caused by repetition reduction introduces additional complexity during the data preprocessing step, and the choice of padding strategy can significantly impact the model's performance. Despite these limitations, the proposed padding method shows that the soft targets can be of arbitrary length, instead of strictly shorter than the original strong targets like our assumption in Chapter 2.

In conclusion, integrating SDTW into a deep neural network model extends the model's capability to handle annotation misalignment, which in turn improves its stability and robustness.

Appendix A

Algorithm Pseudocode

Chapter 2 provides a detailed mathematical formulation of the SDTW algorithm. In this appendix, we present an overview of pseudocode for the forward and backward passes, following the notation and terminology used by Zeitler et al. [12].

A.1 Forward Pass

Algorithm 1 shows the overview of the forward pass of SDTW. A complete row and column of a matrix $\mathbf{D}$ is denoted as $\mathbf{D}(i,:)$ and $\mathbf{D}(:,j)$, respectively. During initialization, we insert an additional row and column at the beginning of the matrix $\mathbf{D}$ and set $\mathbf{D}(0,0) = 0$.

A.2 Backward Pass

Algorithm 2 shows the overview of the backward pass of SDTW. Similar to the forward pass, we append an additional row and column at the end of the matrices. The additional row and column of are corespondingly intialzied with 0 and $-\infty$, except for $\mathbf{C}(M+1, N+1) = 0$, $\mathbf{D}(M+1, N+1) = \mathbf{D}(M, N)$ and $\mathbf{E}(M+1, N+1) = 1$.

Algorithm 1 Forward Pass of SDTW

```
Require: $\mathbf{X}$, $\mathbf{Y}$, $\gamma > 0$
Ensure: Accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{M \times N}$
  $\mathbf{D}(0,0) = 0$
  $\mathbf{D}(i,:) = \mathbf{D}(:,j) = \infty$
  for $i = 1$ to $M$ do
      for $j = 1$ to $N$ do
          Compute local cost: $\mathbf{C}(i,j) = \|\mathbf{x}_i - \mathbf{y}_j\|_2^2$
          Update accumulated cost:
              $$\mathbf{D}(i,j) = \mathbf{C}(i,j) + \min^{\gamma}\left(\mathbf{D}(i-1,j), \mathbf{D}(i,j-1), \mathbf{D}(i-1,j-1)\right)$$
      end for
  end for
      return D
```

Algorithm 2 Backward Pass of SDTW

```
Require: $\mathbf{C}, \mathbf{D}$, $\gamma > 0$
Ensure: Gradient matrix $\mathbf{H} \in \mathbb{R}^{M \times N}$
  $\mathbf{C}(M+1,:) = \mathbf{C}(:,N+1) = 0$
  $\mathbf{D}(M+1,:) = \mathbf{D}(:,N+1) = -\infty$
  $\mathbf{D}(M+1,N+1) = D(M,N)$
  $\mathbf{E}(M+1,:) = \mathbf{E}(:,N+1) = 0$
  $\mathbf{E}(M+1,N+1) = 1$
  for i = M down to 1 do
      for j = N down to 1 do
          Compute $\mathbf{F}^{1,0}(i,j), \mathbf{F}^{0,1}(i,j), \mathbf{F}^{1,1}(i,j)$
          Update $\mathbf{E}(i,j) = \mathbf{E}(i+1,j)\mathbf{F}^{1,0}(i,j) + \mathbf{E}(i,j+1)\mathbf{F}^{0,1}(i,j) + \mathbf{E}(i+1,j+1)\mathbf{F}^{1,1}(i,j)$
          Compute $\mathbf{G}(i,j) = \frac{\partial \mathbf{D}(i,j)}{\partial \mathbf{C}(i,j)}$
          Update $\mathbf{H}(i,j) = \mathbf{E}(i,j)\mathbf{G}(i,j)$
      end for
  end for
      return H
```

Appendix B

Model Template Weight

Figure B.1 shows the weight distribution of the final layer over two chord classes. We enable weight sharing across twelve chromas to counteract the small dataset size, Hence, the weight distribution appears similar to a transposition of a single chord. The final weight of the major chord class is more distinguished and vivid than that of the minor chord class, which is fainter, indicating less weight. This behaviour is also reflected in the original dChord model [13], where, due to the imbalance of the major and minor chords in the dataset, the model tends to favor major chords over minor chords.

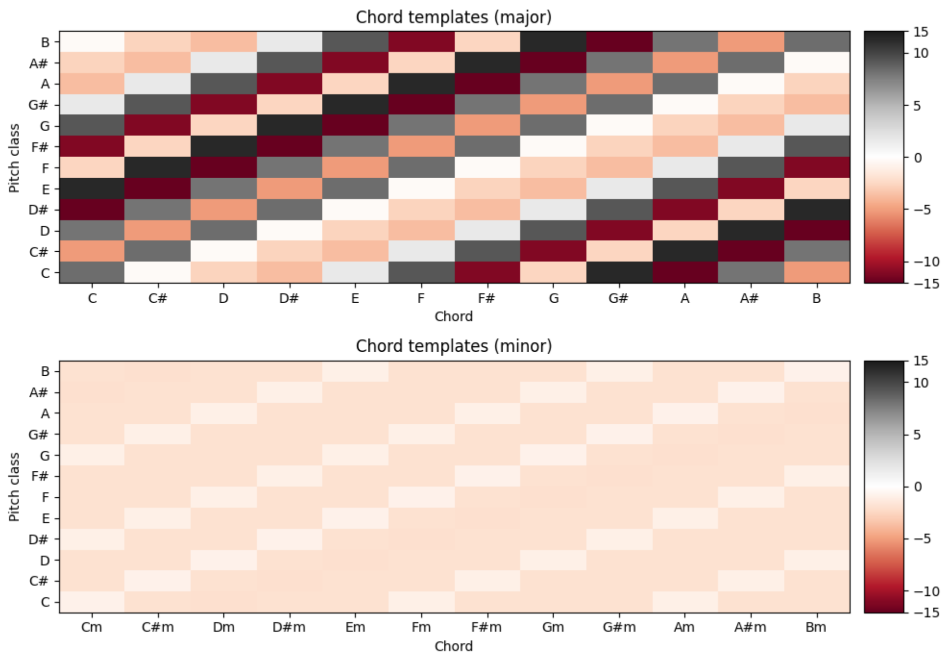


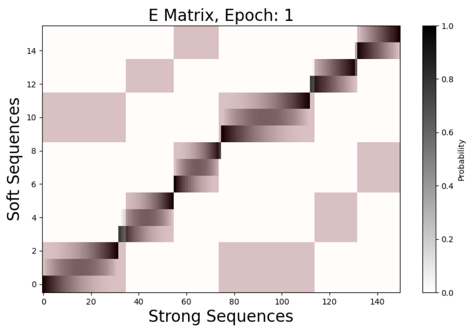
Figure B.1: Weight distribution of the final layer over two chord classes. Higher values indicate larger weight.

Appendix C

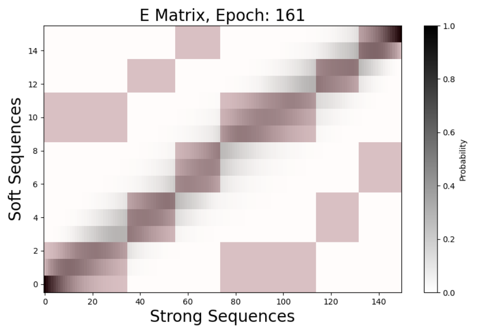
Soft Temperature's Effect

In Section 4.3, we observe that the gradient distribution along the alignment path appears to be blurry and not exclusively concentrated along the alignment path for $\gamma = 0.3$. Therefore, we investigate the effect of different soft temperatures γ on the gradient distribution during training. We choose two extreme values of γ, which are 0.001 and 10, to observe the difference in gradient behaviour. For this experiment, we use the configuration of soft length 16 with uniform padding from Section 4.2.

Figure C.1 shows the gradient behaviour with different soft temperatures γ of 0.001 and 10. At epoch 161, we see that higher γ leads to a more diffuse gradient distribution along the alignment path, while lower γ results in a sharper gradient distribution at the first epoch. This behaviour is consistent with the findings in [3].



(a) $\gamma = 0.001$, Epoch 1



(b) $\gamma = 10$, Epoch 161

Figure C.1: Gradient with different soft temperature γ. Soft length is 16 with uniform padding.

Bibliography

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, *Automatic Music Transcription: An Overview*, IEEE Signal Processing Magazine, 36 (2019), pp. 20–30.

[2] A. L. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, *Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings*, in ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 3852–3856.

[3] M. Cuturi and M. Blondel, *Soft-DTW: a Differentiable Loss Function for Time-Series*, 2018.

[4] S. Damadi, G. Moharrer, and M. Cham, *The Backpropagation algorithm for a math student*, 2023.

[5] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks*, ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning, 2006 (2006), pp. 369–376.

[6] C. Harte, M. B. Sandler, S. Abdallah, and E. Gómez, *Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations*, in Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), London, UK, 2005, pp. 66–71.

[7] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, *Crepe: A Convolutional Representation for Pitch Estimation*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 161–165.

[8] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, 2017.

[9] M. Müller, *Fundamentals of Music Processing – Audio, Analysis, Algorithms, Applications*, Springer Verlag, 2015.

[10] M. Müller, *Dynamic Time Warping*, Information Retrieval for Music and Motion, 2 (2007), pp. 69–84.

[11] C. Sekhar and P. Meghana, *A Study on Backpropagation in Artificial Neural Networks*, Asia-Pacific Journal of Neural Networks and Its Applications, 4 (2020), pp. 21–28.

[12] J. Zeitler, M. Krause, and M. Müller, *Soft Dynamic Time Warping with Variable Step Weights*, in ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 356–360.

[13] J. Zeitler and M. Müller, *Preparation Course PyTorch Notebook*, 2025.