Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Internship Report

Report Title
Training with Unaligned Dataset: Soft Dynamic Time Warping

submitted by

Quang Hoang Nguyen Vo

submitted

September 10, 2025

Supervisors

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

AUDIO LABS

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

Fraunhofer IIS

Abstract

The evolution of Deep Neural Networks (DNNs) has shifted the paradigm of music information retrieval (MIR) 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 (dDTW) through the soft-DTW (sDTW) algorithm when training with weakly aligned data. The main objective is to integrate soft-DTW as a loss function in the training process of a template-based chord recognition model. The dataset will have its chord label timestamps distorted or removed to simulate weakly or unaligned data. The dDTW loss function will then be used to train the model with the distorted dataset. The results will be compared with those obtained using the original dataset and the Connectionist Temporal Classification (CTC) loss function.

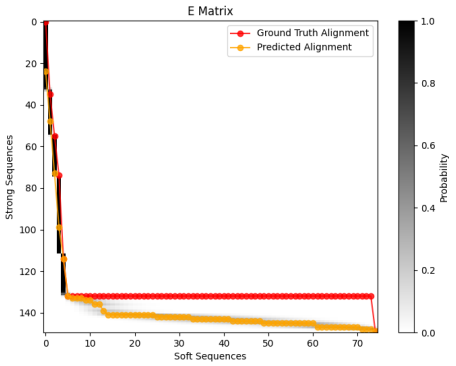


Figure 1: Example of the alignment cost matrix computed by the soft-DTW algorithm between a prediction sequence and a target sequence. The red line indicates the optimal warping path. The orange line is the predicted path.

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 [1], audio embeddings [2], automatic music transcription [3].

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 ease up 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 [4].

Chapter 2

Soft Dynamic Time Warping Algorithm

In this chapter, we present the mathematical formulation of the soft-DTW algorithm.

2.1 Definition and Notation

Let $\mathbf{x} = (x_0, \ldots, x_{N-1}) \in \mathbb{R}^N$ and $\mathbf{y} = (y_0, \ldots, y_{M-1}) \in \mathbb{R}^M$ be two time series of representing the sequence of predictions and strong targets, respectively. We then denote the soft target sequence as $\mathbf{y}' = (y'_0, \ldots, y'_{M'-1}) \in \mathbb{R}^{M'}$, where $M' < M$. The objective of soft-DTW is to calculate the alignment cost matrix between $\mathbf{x}$ and $\mathbf{y}'$.

2.2 Forward Pass

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

$$\mathbf{C}(i, j) = \|x_i - y'_j\|^2 \tag{2.1}$$

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

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

where $\gamma > 0$ is a smoothing parameter that controls the softness of the minimum operation, and S is the set of values over which the soft minimum is computed. Combining with the step constraint of DTW, we only consider three possible predecessor cells for each cell (i, j), which

Algorithm 1 Forward Pass of Soft-DTW

Require: Time series $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{y}' \in \mathbb{R}^{M'}$, smoothing parameter $\gamma > 0$
Ensure: Accumulated cost matrix $\mathbf{D} \in \mathbb{R}^{M' \times N}$
  Initialize $\mathbf{D}(0,0) = 0$
  for $i = 1$ to M' **do**
    $\mathbf{D}(i,0) = \infty$
  end for
  for $j = 1$ to N **do**
    $\mathbf{D}(0,j) = \infty$
  end for
  for $i = 1$ to M' **do**
    for $j = 1$ to N **do**
      Compute local cost: $\mathbf{C}(i,j) = \|x_j - y'_i\|^2$
      Update accumulated cost:

$$\mathbf{D}(i,j) = \mathbf{C}(i,j) + \min^{\gamma}(\mathbf{D}(i-1,j), \mathbf{D}(i,j-1), \mathbf{D}(i-1,j-1))$$

    end for
  end for
    return D

are $(i-1,j)$, $(i,j-1)$, and $(i-1,j-1)$. Thus the accumulated cost matrix $\mathbf{D}$ is computed recursively as follows:

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

The recursive algorithm is summarized in Algorithm 1.

2.3 Backward Pass

In order to train the model, we need to compute the gradients of the soft-DTW loss with respect to the model parameters. The backward pass computes the gradient $H \in \mathbb{R}^{M' \times N}$ of the soft-DTW cost with respect to the cost matrix $\mathbf{C} \in \mathbb{R}^{M' \times N}$. 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:

$$\mathbf{H}(i,j) = \frac{\partial \mathbf{D}(M',N)}{\partial \mathbf{D}(i,j)} \frac{\partial \mathbf{D}(i,j)}{\partial \mathbf{C}(i,j)} \tag{2.4}$$

During the backward pass, we consider three predecessor cells for each cell (i,j), which are $(i+1,j)$, $(i,j+1)$, and $(i+1,j+1)$. This expands the gradient computation to:

$$\mathbf{H}(i,j) = \frac{\partial \mathbf{D}(M',N)}{\partial \mathbf{D}(i+1,j)} \frac{\partial \mathbf{D}(i+1,j)}{\partial \mathbf{D}(i,j)} + \frac{\partial \mathbf{D}(M',N)}{\partial \mathbf{D}(i,j+1)} \frac{\partial \mathbf{D}(i,j+1)}{\partial \mathbf{D}(i,j)} + \frac{\partial \mathbf{D}(M',N)}{\partial \mathbf{D}(i+1,j+1)} \frac{\partial \mathbf{D}(i+1,j+1)}{\partial \mathbf{D}(i,j)} \tag{2.5}$$

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 $p, q \in \{0,1\}$.

Algorithm 2 Backward Pass of Soft-DTW

Require: $\mathbf{C}, \mathbf{D} \in \mathbb{R}^{M' \times N}$, $\gamma > 0$
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

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

$$\frac{\partial \min^{\gamma}(s)}{\partial s} = \frac{e^{-s/\gamma}}{\sum_{s' \in \mathcal{S}} e^{-s'/\gamma}} \tag{2.6}$$

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

$$\mathbf{F}^{1,0}(i,j) = \frac{\partial \min^{\gamma}(\mathbf{D}(i+1,j))}{\partial \mathbf{D}(i,j)} = \frac{e^{-(\mathbf{C}(i+1,j) - \mathbf{D}(i,j))/\gamma}}{e^{-\mathbf{D}(i,j)/\gamma}} \tag{2.7}$$

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

$$\mathbf{E}(i,j) = \mathbf{E}(i+1,j)\mathbf{F}^{1,0}(i,j) + \mathbf{E}(i,j+1)\mathbf{F}^{0,1}(i,j) + \mathbf{E}(i+1,j+1)\mathbf{F}^{1,1}(i,j) \tag{2.8}$$

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:

$$\mathbf{H}(i,j) = \mathbf{E}(i,j)\mathbf{G}(i,j) \tag{2.9}$$

For initialization, we add and extra column and row to $\mathbf{E}$, setting $\mathbf{E}(M'+1, N+1) = 1$ and $\mathbf{E}(i,i)$ to be $-\infty$ and 0 for the rest of the extra row and column. Algorithm 2 presents the complete overview of the recursion for gradient computation.

Chapter 3

Experimental Setup

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

3.1 Dataset

For the experiment, we use the Beatles dataset retrieved from Isophonics [5], 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 reduced the number of chord types to only 24 (12 chromas with their respective major or minor variant)[6]. We split the dataset into training, validation, and test sets. For test set, a short segment of Let It Be is used, while the rest of the dataset is split into 3:1 ratio for training and validation.

We choose 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, effectively reducing its length by around 85% on average (see Figure 3.1).

However, this method introduce a problem where batching target sequences of different lengths is not possible. To address this issue, after reduction, we pad the sequences 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 to repeating all elements in the sequence or "stretching" the sequence to the desired length, whereas last element only repeats the last element until the desired length is reached. After some experiments, we found that a soft length of 16 covers all of possible reduced sequences while keeping the reduction ratio high.

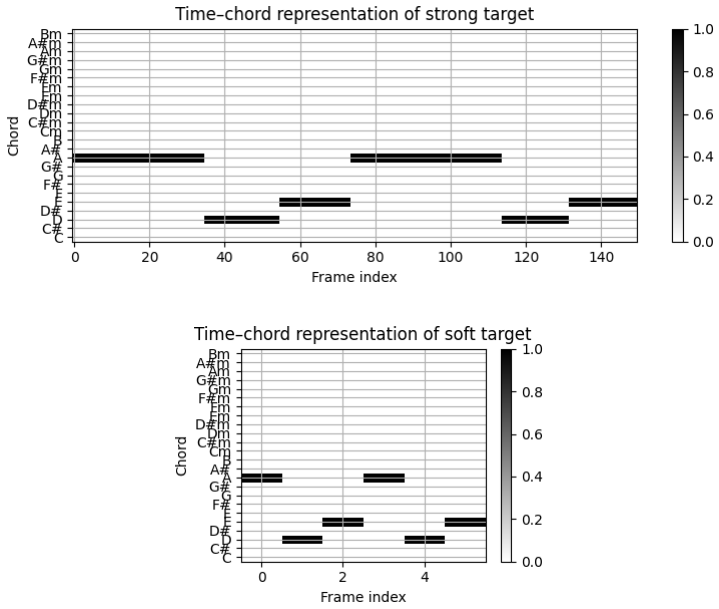


Figure 3.1: Example of strong and soft target sequences. The strong target sequence has a length of 150, while the soft target sequence has a length of 5 after removing adjacent repetitions.

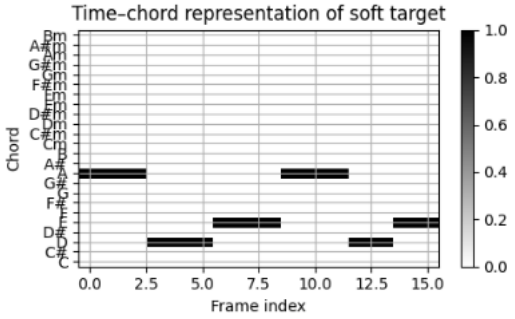
Layer	Input Dimension	Output Dimension	Parameters
Log-compression	(12, 150)	(12, 150)	1
Normalization	(12, 150)	(12, 150)	0
dChord	(12, 150)	(24, 150)	24
softmax	(24, 150)	(24, 150)	0

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

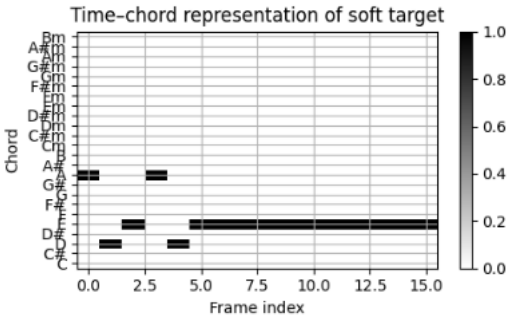
3.2 Model Architecture

Given the aim of this experiment is to evaluate the performance of the proposed SDTW loss function, the network architecture plays a minor role and are kept simple. Therefore, we used a simple chord recognition network (dChord) that based on the template-based chord recognition algorithm. 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 illustrate the components of the architecture with their respective input and output dimensions.

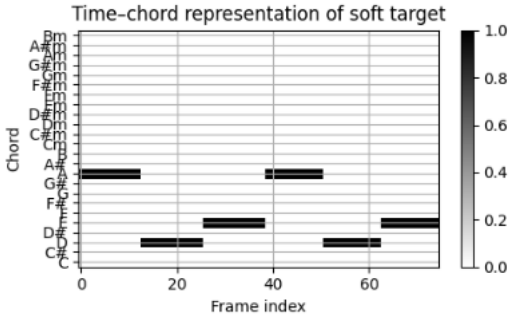
During the training process, we use Adam Optimizer [7] with a a learning rate $\alpha = 0.1$ with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Since the dataset is small of only 43 training segments, we use a single batch of 43 targets train the model over 200 epochs for soft targets and 50 epochs for strong targets. We also enables weight sharing, where the weights of the chords in the same chord class (major or minor) are share, 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.



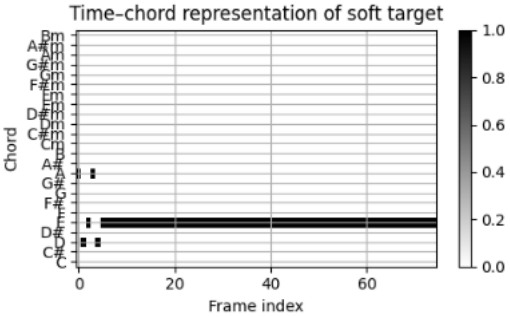
(a) Uniform padding with soft length of 16.



(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.

Chapter 4

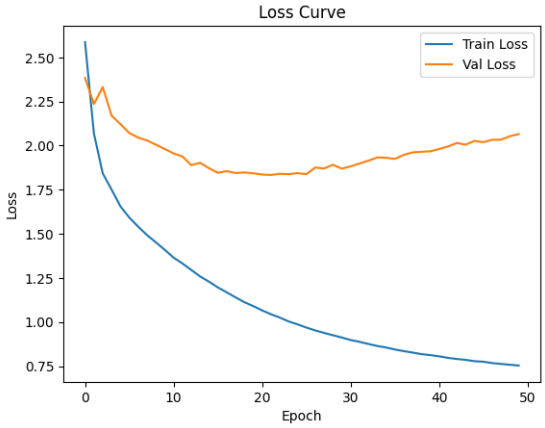
Evaluation

In this section, we evaluate the performance of the aforementioned soft-DTW loss function against the baseline model trained with binary cross-entropy loss for the task of chord recognition. We train the each model 5 times to observe any randomness present in the training process, to which the model consistently achieves similar performance metrics accross multiple runs. We train the models on a total of 5 cases, with a baseline case of strong taget and 4 soft cases with 2 soft length of 16 and 75, each with 2 padding strategies (uniform and last element repetition).

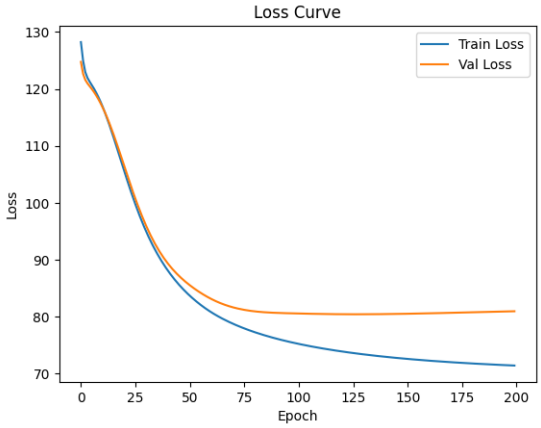
4.1 Training Curve

Figure 4.1 shows the training and validation loss curves of the models trained on strong and soft targets. We can see overall the loss decreases over time, indicating that the models are learning from the data, despite much higher loss value for the model trained on soft target, which ranges from 130 to 70, compared to 2.5 to 0.75 for the model trained on strong target. This is expected, as the soft target contains less information, making it more difficult for the model to learn. In all cases, we observe the overfitting behaviour after one-third of the training processes (epoch 20 for strong target, epoch 60 for soft targets), where the validation loss starts to increase while the training loss continues to decrease.

Figure 4.2 compares the training curves among different soft target configuration. Interestingly, three of them have similar curve shapes, indicating that the model's performance is relatively stable across different configurations. One outlier is the last element repetition case of soft length 75, which different flow in validation loss since even the first several epochs. Furthermore, we can see that longer padding length tends to facilitate overfitting as evidenced by the discrepency of the loss between training and validation in case of soft length 75 compared to soft length 16, both with uniform padding. Meanwhile, in soft length 16 case, higher difference is shown in last element repetition compared to uniform padding. This means that the choice of padding strategy and soft target length can from slightly to significantly impact the model's learning process, with certain configurations being more prone to overfitting than others. From this results, we see that uniform padding combined with shorter soft length results in the best performance.



(a) Training and validation loss curves of the model trained on strong target with binary cross-entropy loss. The model converges after around 20 epochs.



(b) Training and validation loss curves of the model trained on soft target with soft-DTW loss. The model converges after around 60 epochs.

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

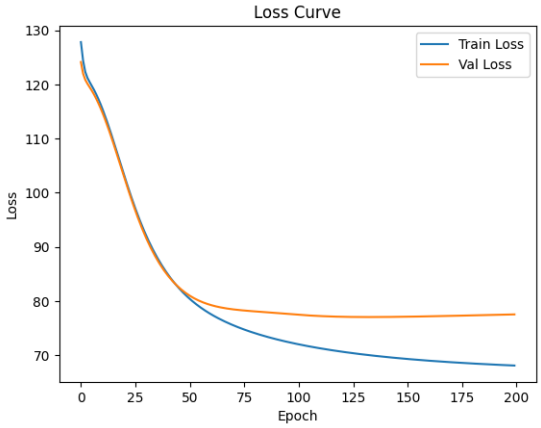
Target/ Original Length	Loss Function (Pad Strategy)	Test F1-score (%)	Epochs to best val. loss
150/150	Binary Cross-entropy	0.765	50
16/150	Soft-DTW (uniform)	0.761	200
75/150	Soft-DTW (uniform)	0.761	200
16/150	Soft-DTW (last)	0.761	200
75/150	Soft-DTW (last)	0.681	200

Table 4.1: Results of the baseline model trained on strong target

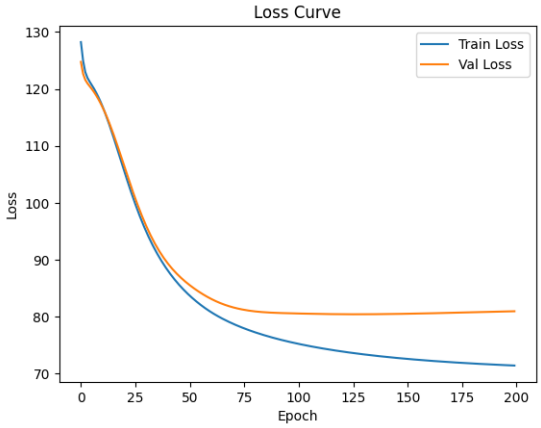
4.2 Training results

Table 4.1 summarizes the results of the baseline model trained on strong target with binary cross-entropy loss and the model trained on soft target with soft-DTW 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.

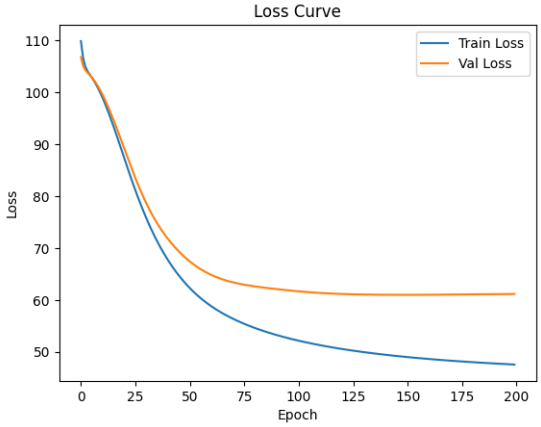
Overall, we see that sDTW loss is able to achieve comparable performance to the baseline model trained on strong target, with a slight decrease in F1-score of 0.4%. This shows that sDTW is able to effectively learn from the soft target that contains less or noisy information, making it more robust to variations in the input sequence. Observing the results of different padding strategy, we find that last element repetition leads to similar performance as uniform padding for short soft lengths though it eventually degrades as the soft length increases. This is likely due to the repetition, which introduces a bias towards the last element, making it more prominent the longer the soft length. Figure **??** shows different training target sequences with respective padding strategy and soft length. We can see that uniform padding distributes the repetitions evenly across all elements, while last element repetition creates a long tail of the last element. We speculate this would cause the model to overfit to the last element, as it becomes more



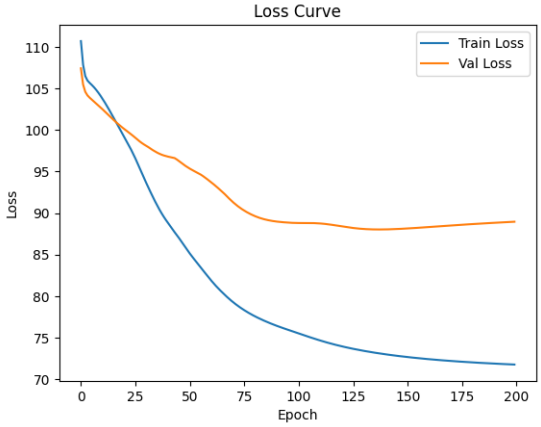
(a) Uniform padding with soft length of 16.



(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 4.2: Comparison between different soft target configuration

frequent in the training data; therefore, it undermines the model's ability to generalize even with sDTW loss. Nonetheless, it also shows that sDTW are rather flexible, being able to compensate for small mistakes in annotation. And it would perform just as well for longer length had the padding was uniform.

4.3 Model Template Weight

In order to understand the behavior of the model trained with soft targets, we analyze the weight distribution of the model parameters. Specifically, we examine the distribution of the weights in the final layer of the model, which is responsible for making the predictions.

Figure 4.3 shows the weight distribution of the final layer over two chord classes. As mentioned before, 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. It is apparent that the final weight of the major chord class is more distinguished and vivid than that of the minor

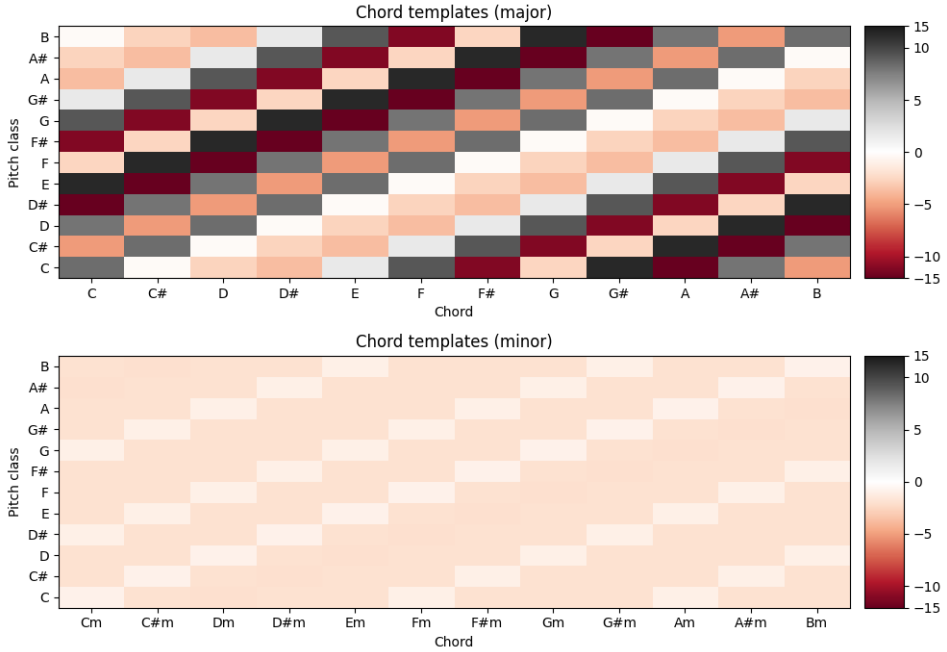


Figure 4.3: Weight distribution of the final layer over two chord classes. Higher values indicate larger weight.

Chord Class	Number of Samples (strong)	Percentage (strong)	Number of Samples (soft length 16)	Percentage (soft length 16)
Major	5843	90.59%	613	89.10%
Minor	607	9.41%	75	10.90%
No Chord	0	0%	0	0%
Total	6450	100%	688	100%

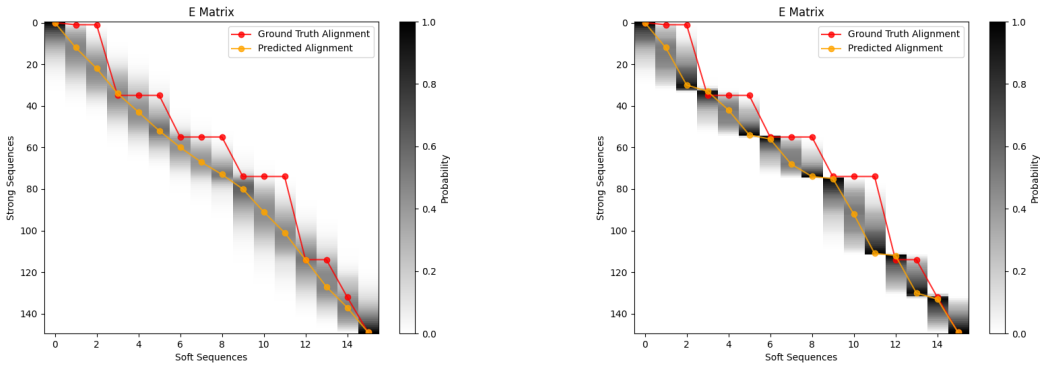
Table 4.2: Distribution of chord classes in the dataset.

chord class, which is more faint, indicating less weight. This is mainly due to class imbalance presents in our dataset. Taking a closer look in Table 4.2, we can see that the major chord class is more prevalent in the dataset, taking up 91% of the total chord contained in strong target case.

In the case of soft targets, we observe a less pronounced class imbalance, with the minor chords now taking up to 12% of the total chord in soft target case. Further investigation with different soft lengths using uniform padding strategy shows that the class frequency fluctuates with respect to the soft length, albeit shows a more balanced distribution compared to the strong target case as shown in Table 4.3. In conclusion, repetition reduction for creating soft target may have a small to noticable effect on class balance, but in general, the model have a bias toward major chords due to minor chords being underrepresented in the dataset.

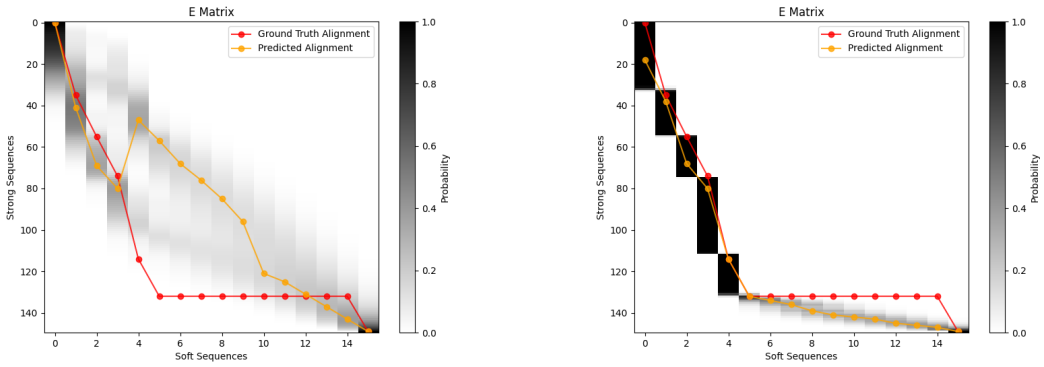
Chord Class	Soft "0"		Soft 16		Soft 75	
	Samples	%	Samples	%	Samples	%
Major	316	87.78%	613	89.10%	2656	88.53%
Minor	44	12.22%	75	10.90%	344	11.47%
No Chord	0	0%	0	0%	0	0%
Total	360	100%	688	100%	3000	100%

Table 4.3: Distribution of chord classes in the dataset with different soft lengths. "0" refers to no padding. Padding strategy is uniform repetition.



(a) Epoch 1 (b) Epoch 161

Figure 4.4: Gradient over epochs with soft length 16 and uniform padding.



(a) Epoch 1 (b) Epoch 161

Figure 4.5: Gradient over epochs with soft length 16 and last element repetition padding.

4.4 Gradient Analysis

Looking at the computed gradients during training, as shown in Figure 4.4, with different soft lengths and uniform padding, we can observe the change in probability distribution of the alignment path. We notice that at first the gradient assumes 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. This shows 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 4.5. In such cases, the gradient behaviour becomes more pronounced, as it quickly converges to the ground truth alignment from initial diagonal prior. Despite these cases cause a large drop in model accuracy, we can see the flexibility of sDTW loss, being able to adapt to different alignment paths.

Chapter 5

Conclusions

In this report, we have presented a comprehensive study on the use of soft dynamic-time-warping 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 the unseen data, even if the training data has missing or imprecise temporal information, with a small degree of degradation on performance. Our experiments provide a method to work around the batching limitations when dealing with variable-length sequences, and we have shown that the model trained with soft targets is able to generalize 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, which may or may not contain imperfections regarding the timing and duration of musical events.

Furthermore, our analysis of the model's behavior reveals interesting insights into the learning process. The gradient flow during training shows that sDTW is able to effectively find the time warping alignment between the input and target sequences, mitigating the impact of misalignments and improving the overall robustness of the model.

Several limitations of our approach should be acknowledged. First, the small and imbalance data set used for the experiment may not fully capture the diversity of musical styles and chord progression. Most notable is the lack of representation of minor chords, which causes the model to be less effective in recognizing these 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 data preprocessing step, and the choice of padding strategy can significantly impact the model's performance.

In conclusion, integrating sDTW into deep neural network model extends the model's capability to handle misalignment, hence improve its stability and robustness.

Bibliography

[1] J. Kim, J. Salamon, P. Li, and J. Bello, "Crepe: A convolutional representation for pitch estimation," pp. 161–165, 04 2018.

[2] A. L. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," pp. 3852–3856, 2019.

[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] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks," vol. 2006, pp. 369–376, 01 2006.

[5] C. Harte, M. B. Sandler, S. Abdallah, and E. Gómez, *Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations*. PhD thesis, London, UK, 2005.

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

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.