HARMONY TRANSFORMER: INCORPORATING CHORD SEGMENTATION INTO HARMONY RECOGNITION

Tsung-Ping Chen and Li Su
Institute of Information Science, Academia Sinica, Taiwan
{tearfulcanon, lisu}@iis.sinica.edu.tw

ABSTRACT

Musical harmony analysis is usually a process of unfolding and interpreting the hierarchical structure of music. Computational approaches to such structural analysis are still challenging, owing to the fact that the boundary between different harmonic states (such as chord functions) is not explicitly defined in the audio or symbolic music data. It is a novel approach to improve chord recognition by jointly identifying chord change using end-to-end sequence learning. In this paper, we propose the Harmony Transformer, a multi-task music harmony analysis model aiming to improve chord recognition through incorporating chord segmentation into the recognition process. The integration of chord segmentation and chord recognition is implemented with the Transformer, a deep sequential learning model yielding fruitful results in the field of natural language processing. A non-autoregressive decoding framework is also adopted here in aid of concatenating the two highly correlated tasks. Experiments of both chord symbol recognition and functional harmony recognition on audio and symbolic datasets demonstrate that explicitly learning the hierarchical structural information of musical data can facilitate and improve the harmony recognition.

1. INTRODUCTION

Automatic chord recognition is a hallmark research topic in the field of music information retrieval (MIR) and has been widely studied. In the past decade, this problem has been dealt with by using a variety of deep learning methods ranging from multi-layer perceptrons (MLPs) [1] to more complex models such as convolutional neural networks (CNNs) [2–6] and recurrent neural networks (RNNs) [7–11]. Hybrid systems such as convolutional recurrent neural network (CRNN) are also introduced to the chord recognition task for taking advantage of different neural networks [12–15].

In spite of the significant achievements for deep learning models to advance the state of the art, further improvement in the chord recognition task appears to be limited and has reached a glass ceiling [4]. Such limitations may result from the incompetence for deep learning methods to infer the hierarchical structure of music based on frame-level data. An overlooked tendency in many previous studies is that the majority of chord recognition models regard a music piece as a concatenation of semantically incomplete segments, and therefore are ineffective to capture complex musical grammar such as chord transition [13]. Figure 2 (see Section 4.4) depicts an example where an RNN model cannot segment chord boundaries adequately when arpeggios or key modulation exists. This problem was dealt with by integrating temporal knowledge such as the metric and beat information [16, 17], by adding post-processing such as hidden Markov models (HMM) [18, 19], or by feature engineering such as the harmonic change detection function (HCDF) [20] and beat-synchronous audio features [21]. One recent study attempted to solve this problem by incorporating models which recognize chords at different levels, i.e., the frame level and the chord level, and the cooperative models gained improved chord recognition results [14].

In this paper, we propose the Harmony Transformer, which jointly integrates chord recognition and chord segmentation into an end-to-end chord recognition system. The Harmony Transformer is based on the Transformer, a deep neural network nowadays applied to a wide variety of sequence modeling problems such as machine translation [22], natural language understanding [23], music generation [24], and so forth. For the encoder-decoder architecture of the Transformer, we assign the chord segmentation task to the encoder, and the chord recognition task to the decoder. This division allows the chord recognition network to benefit from the prediction of the chord segmentation. In comparison to other designs using RNN, which is considerably time-consuming owing to heavy sequential computations, the Transformer is more effective in speed, as it enforces parallelization to process sequential data. The Harmony Transformer is also capable of a complex analytic scenario—harmonic function [1] recognition [25]. Our experiment results highlight the advantage of the proposed model to improve both the chord symbol recognition and the harmonic function recognition. To the best of our knowledge, this is the first work that connects the chord segmentation task with the chord recognition task through the Transformer framework. We hope

[1] 'Harmonic function' refers to the diatonic function in music which denotes the relationship of a chord to a tonal center.

that our contribution facilitates more developments in automatic chord recognition.

2. HIERARCHICAL STRUCTURES AND CHORD RECOGNITION

The chord recognition can be formulated as a *sequence labeling* task [26] that assigns a categorical label (e.g., chord name) to each element of a given sequence (e.g., musical sequence). Considering the sequential and structural nature of music, chord recognition task resembles many natural language processing (NLP) problems, in such a way that both the sequential arrangements of chords in music and words in language are dominated by higher-level rules—the function of harmony and the grammar in language. A compelling recognition model for music or language is, therefore, expected to unfold the intrinsic hierarchical structure of the sequences. A typical example is the part-of-speech (POS) tagging task in NLP, which assigns a specific part of speech to each word in a sentence. Albeit many words can represent more than one part of speech at different times, deep learning models have shown their ability to regulate local-level assignments with a given context, and demonstrated satisfactory performance. [2]

On the other hand, the chord recognition results in recent years still could not be satisfactory [1,8,13–15]. A primary difference between language and music is that each word in the input sentence of the POS tagging task represents a semantically meaningful unit, whereas the musical segments to be labeled in the chord recognition task are not necessary harmonically completed. It can be argued that the unsatisfying achievements in the chord recognition task can be associated with the deficiency of explicit knowledge of the harmonic boundaries within a musical sequence. Since the sequential property and the hierarchical structure are shared characteristics between language and the harmonic progression, it would be advantageous to introduce the segmentation information of harmony in higher-level musical hierarchy into deep learning models of chord recognition.

Recent research tackled the hierarchical issue of frame-wise chord recognition using two RNN-based models to separately predict the duration and the transition of each chord in a *parallel* fashion [14]. Concretely, frame-wise chord predictions are first generated through a CNN-based *acoustic model*, which are then simultaneously fed into the two RNN-based models—the *chord duration model* and the *harmonic language model* to predict the time points of chord change at the frame level and the chord progression at the chord level respectively. The two RNN-based models are trained separately from the acoustic model, and are utilized as *language model* which computes the probability distribution of each token sequentially.

It has been demonstrated that incorporating musical hierarchy is promising to improve chord recognition. Nevertheless, the local assignment of chord labels in practice is dependent *successively* on the knowledge from different hierarchical levels. We hence argue that 1) the chord change and the chord progression should be modeled in a vertical rather than a horizontal manner; in other words, we suppose that the chord recognition problem belongs to a higher level in the musical hierarchy than the chord segmentation problem, and the recognition of chords should base on the result of segmentation. 2) The models processing higher-level information such as the two RNN-based language models should be trained *jointly* with the lower-level ones. In this study, we propose a new framework that predicts chord progression for a given sequence according to the chord segmentation result, and is trained in an end-to-end manner.

[2] The per-token accuracy of the POS tagging task had surpassed 96% in 2000. For an overview of the state of the art of the POS tagging task, see: `https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art)`.

3. THE HARMONY TRANSFORMER

The proposed model, as shown in Figure 1, follows the encoder-decoder architecture of the Transformer [22], while adopting a non-autoregressive framework for treating segmentation as the intermediate before the recognition process [27]. The encoder of the Harmonic Transformer performs the chord segmentation task from the input data sequence (e.g., chroma), and the decoder performs the chord recognition task from the input data sequence along with the segmentation result from the encoder.

3.1 Basic units

The Harmony Transformer model is built on two computational units, that is, *multihead-head attention* (MHA) and *feed-forward network* (FFN). The MHA unit takes three inputs $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, which stand for *queries*, *keys*, and *values*, respectively. The three inputs are matrices with the first dimension representing the number of time steps and the second dimension being the number of feature; the feature sizes of the three inputs are usually the same, while their number of time steps may be different.

In general, the MHA unit computes a weighted sum of $\mathbf{V}$ based on the similarity between $\mathbf{Q}$ and $\mathbf{K}$. To be more exact, the inputs are first 'transformed' through an *attention* mechanism which outputs a *head* (denoted as $\mathbf{D}$). The MHA unit is then constructed by concatenating multiple heads out of several attention mechanisms:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d}}\right)\mathbf{V}\,, \quad (1)$$

$$\mathbf{D}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^{\mathrm{Q}}, \mathbf{K}\mathbf{W}_i^{\mathrm{K}}, \mathbf{V}\mathbf{W}_i^{\mathrm{V}})\,, \quad (2)$$

$$\mathrm{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathbf{D}_1,\cdots,\mathbf{D}_h)\mathbf{W}^{\mathrm{D}}\,, \quad (3)$$

where h is the number of heads, $\mathbf{D}_i$ is the ith head, and d is the dimension of feature. $\mathbf{W}^{\mathrm{D}} \in \mathbb{R}^{d\times d}$, and $\mathbf{W}^{\mathrm{Q}}$, $\mathbf{W}^{\mathrm{K}}$, $\mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{d\times \frac{d}{h}}$ are all learnable parameter matrices. Note that if $\mathbf{Q} = \mathbf{K} = \mathbf{V}$, such attention mechanism is called *self-attention* as the three inputs stand for the same sequence; if $\mathbf{Q} \neq \mathbf{K} = \mathbf{V}$, such attention mechanism is called *encoder-decoder attention* in the case when $\mathbf{K}$ and $\mathbf{V}$ come from the encoder and $\mathbf{Q}$ comes from the decoder.

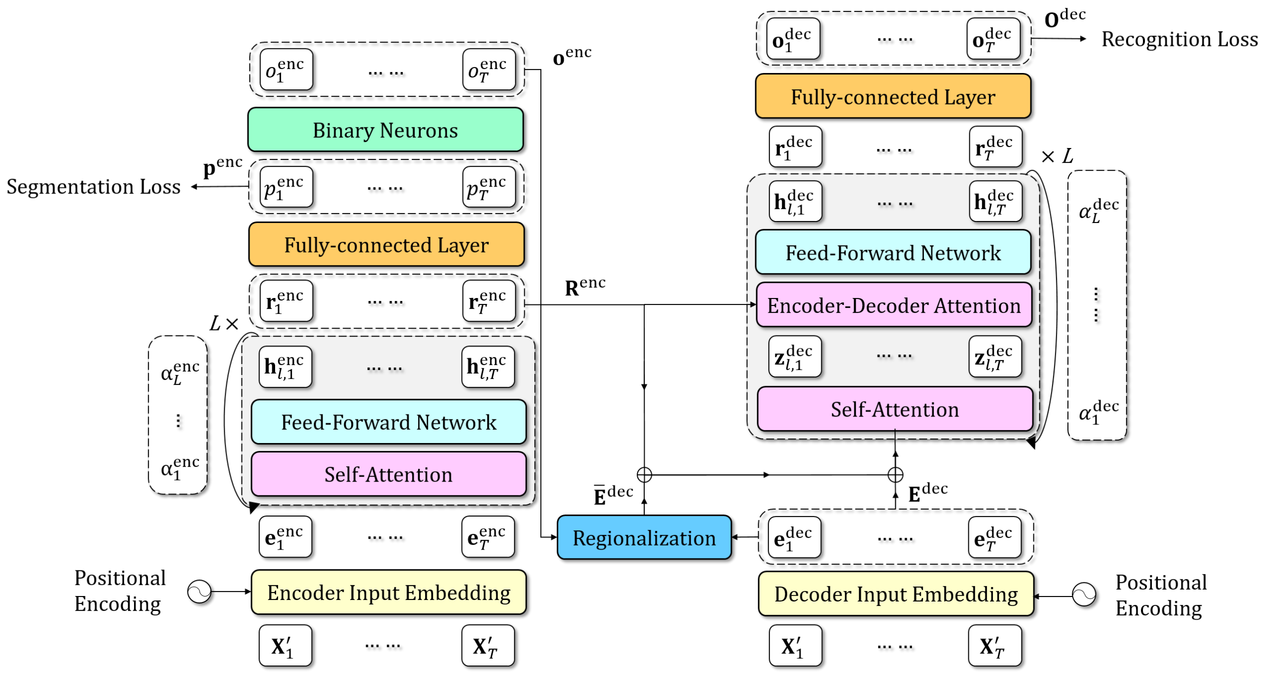

Figure 1. The architecture of the Harmony Transformer. During training, the chord change prediction $\mathbf{p}^{\text{enc}}$ from the encoder is used to calculate the segmentation loss, while the decoder output $\mathbf{O}^{\text{dec}}$ is used to calculate the recognition loss.

On the other hand, the FFN unit is constructed by two fully-connected layers parameterized by weighted matrices and bias vectors $(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2)$ and one activation function in between the two layers:

$$\mathrm{FFN}(\mathbf{X}) = \mathrm{ReLU}(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\,, \qquad (4)[3]$$

where $\mathbf{X}$ is also a matrix like $\mathbf{Q}$, $\mathbf{K}$, or $\mathbf{V}$, and $\mathrm{ReLU}(\mathbf{X}) := \max(0, \mathbf{X})$ is the element-wise rectified linear unit activation. In practice, both the MHA and the FFN computations include residual connections [28] and layer normalization [29]:

$$\mathrm{MHA}^*(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{LayerNorm}(\mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) + \mathbf{Q})\,,$$
$$\mathrm{FFN}^*(\mathbf{X}) = \mathrm{LayerNorm}(\mathrm{FFN}(\mathbf{X}) + \mathbf{X})\,.$$

In the rest of the paper, we omit the $*$ symbol for simplicity. More detailed descriptions of MHA and FFN can be found in Section 3.2 and Section 3.3 of [22].

3.2 Input embedding

We denote the sequence of frame-level features (e.g., piano roll or chromagram) with T time steps as $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_T]^{\mathrm{T}}$,[4] in which $\mathbf{x}_t$ is a feature vector of $\mathbf{X}$ at time t. The data representation entering the Harmony Transformer at time t, as denoted by $\mathbf{X}'_t$ here, is a segment of $\mathbf{X}$ around t: $\mathbf{X}'_t := [\mathbf{x}_{t-\tau}, \cdots, \mathbf{x}_{t+\tau}]^{\mathrm{T}}$, where the length of the segment is $2\tau+1$. For the embedding process, the encoder (or decoder) maps each $\mathbf{X}'_t$ into an encoder (or decoder) embedding through MHA and FFN units, then flattens it into a vector $\mathbf{e}^{\text{enc}}_t$ (or $\mathbf{e}^{\text{dec}}_t$). More specifically, the embedding vector $\mathbf{e}_t$ which is to be fed into time step t of either the encoder or the decoder is:

$$\mathbf{e}_t = \mathrm{Flatten}(\mathrm{FFN}(\mathrm{MHA}(\mathbf{X}'_t, \mathbf{X}'_t, \mathbf{X}'_t))\mathbf{W}^{\mathrm{e}}), \qquad (5)$$

where $\mathbf{W}^{\mathrm{e}}$ is the parameter matrix to be learned, and the encoder and the decoder use different set of parameters. The embedding sequences for the encoder and the decoder are then $\mathbf{E}^{\text{enc}} = [\mathbf{e}^{\text{enc}}_1, \cdots, \mathbf{e}^{\text{enc}}_T]^{\mathrm{T}}$ and $\mathbf{E}^{\text{dec}} = [\mathbf{e}^{\text{dec}}_1, \cdots, \mathbf{e}^{\text{dec}}_T]^{\mathrm{T}}$, respectively.

Moreover, since the computational units mentioned in Section 3.1 are intrinsically unaware of the sequential order of their inputs, the positional information is added to the feature embeddings. In this work, we adopt the absolute positional encoding composed of sinusoidal functions, as used in [22]:

$$\mathrm{PE}_{t,i} = \begin{cases} \sin(t/10000^{\frac{i}{d}}) \text{ if } i \text{ is even,} \\ \cos(t/10000^{\frac{i}{d}}) \text{ if } i \text{ is odd,} \end{cases} \qquad (6)$$

where $t = 1, \ldots, T$ is the time step, d is the embedding size, and $i = 1, \ldots, d$ is the index of the embedding dimension. And $\mathrm{PE}_{t,i}$ is added into the ith element of $\mathbf{e}_t$.

3.3 Encoder: chord segmentation

The encoder is composed of L layers, each of which comprises a self-attention MHA unit and a FFN unit. Inspired by [30], L softmax-normalized parameters are introduced to the encoder for weighting the hidden states of each layer. Formally, for the embedded sequence of the encoder, $\mathbf{E}^{\text{enc}}$, the encoder computes hidden states

[3] Note that the two additions in the equation are element-wise additions which will 'broadcast' the bias vectors to the same dimensions as the matrices to be added.

[4] In our notation, the normal T means the transpose of a matrix, while the italic T indicates the number of time steps.

$\mathbf{H}_l^{\text{enc}} := [\mathbf{h}_{l,1}^{\text{enc}}, \cdots, \mathbf{h}_{l,T}^{\text{enc}}]^{\text{T}}$ at layer l, where $\mathbf{h}_{l,t}$ denotes the output at time t in the lth layer:

$$\begin{aligned} \mathbf{H}_l^{\text{enc}} &= \text{FFN}(\text{MHA}(\mathbf{H}_{l-1}^{\text{enc}}, \mathbf{H}_{l-1}^{\text{enc}}, \mathbf{H}_{l-1}^{\text{enc}}))\,, \\ \mathbf{H}_0^{\text{enc}} &= \mathbf{E}^{\text{enc}}\,. \end{aligned} \tag{7}$$

The hidden states of all layers are then weighted by the softmax-normalized parameters $\boldsymbol{\alpha}^{\text{enc}} := \{\alpha_l^{\text{enc}}\}_{l=1}^{L}$ to produce a weighted-sum representation $\mathbf{R}^{\text{enc}} := [\mathbf{r}_1^{\text{enc}}, \cdots, \mathbf{r}_T^{\text{enc}}]^{\text{T}}$ with $\mathbf{r}_t^{\text{enc}} = \sum_{l=1}^{L} \alpha_l^{\text{enc}} \mathbf{h}_{l,t}^{\text{enc}}$, which is later taken as one of the inputs of the decoder. The purpose of using the weighted sum of hidden states from all layers instead of using the output of the last layer is that different layers tend to encode specific information which individual tasks may rely on to different extents [30–33].

To predict chord change, $\mathbf{R}^{\text{enc}}$ is then fed into a fully-connected layer followed by a sigmoid activation. That is, the likelihood of chord change at time t is estimated by the equation: $p_t^{\text{enc}} = \text{sigmoid}(\mathbf{w}^{\text{T}}\mathbf{r}_t^{\text{enc}})$, where $\mathbf{w}$ are the parameters to be learned for mapping each $\mathbf{r}_t^{\text{enc}}$ in $\mathbf{R}^{\text{enc}}$ into a real number. And the *segmentation loss* is calculated with $\mathbf{p}^{\text{enc}} := \{p_t^{\text{enc}}\}_{t=1}^{T}$ during training, as depicted in Figure 1. In order to make use of the chord change prediction for the later part of the chord recognition task, we utilize deterministic binary neurons (DBNs) [34, 35] to binarize the real-valued chord change probabilities with hard thresholding. Accordingly, the final output of the encoder $\mathbf{o}^{\text{enc}} := \{o_t^{\text{enc}}\}_{t=1}^{T}$ is a sequence of binary chord segmentation prediction, which is 1 at the point of chord change, or 0 otherwise: $o_t^{\text{enc}} = 1$ if $p_t^{\text{enc}} > 0.5$ and $o_t^{\text{enc}} = 0$ if $p_t^{\text{enc}} \leq 0.5$. That means, when $o_t^{\text{enc}} = 1$, there is a chord change at time t. For example, for a source sequence of 6 segments, $\mathbf{o}^{\text{enc}} = [1, 0, 0, 1, 1, 0,]$ means that there are three *chord regions* with the first region containing three segments, the second containing one segment, and the final containing two segments.

3.4 Decoder: segmentation-informed chord recognition

Similar to the encoder, the decoder also consists of L layers, while in each layer, there is an additional encoder-decoder attention module besides the MHA and the FNN modules to imitates the classical attention mechanism in sequence-to-sequence models [36, 37]. Different from the encoder, the input of the decoder is derived from three sources: $\mathbf{E}^{\text{dec}}$, $\mathbf{o}^{\text{enc}}$, and $\mathbf{R}^{\text{enc}}$. First of all, the embedded sequence of the decoder $\mathbf{E}^{\text{dec}}$ is *regionalized* in line with $\mathbf{o}^{\text{enc}}$ to generate $\bar{\mathbf{E}}^{\text{dec}}$. Precisely, let $\mathbf{c} := \{c_k\}_{k=1}^{K}$ be the K time steps where chord changes, i.e., $o_{c_k}^{\text{enc}} = 1$, then the regionalization unit in Figure 1 replaces each member $\mathbf{e}_t^{\text{dec}}$ in $\mathbf{E}^{\text{dec}}$ with average pooling:

$$\bar{\mathbf{e}}_t^{\text{dec}} := \frac{1}{c_{k+1} - c_k} \sum_{i=c_k}^{c_{k+1}-1} \mathbf{e}_i^{\text{dec}} \;\; \text{for } t \in [c_k, c_{k+1})\,, \tag{8}$$

and the resulting embedding is $\bar{\mathbf{E}}^{\text{dec}} := [\bar{\mathbf{e}}_1^{\text{dec}}, \cdots, \bar{\mathbf{e}}_T^{\text{dec}}]^{\text{T}}$. Next, the decoder takes the original embedding $\mathbf{E}^{\text{dec}}$, the regionalized embedding $\bar{\mathbf{E}}^{\text{dec}}$, and the weighted-sum representation $\mathbf{R}^{\text{enc}}$ to compute hidden states of the decoder $\mathbf{H}_l^{\text{dec}} = [\mathbf{h}_{l,1}^{\text{dec}}, \cdots, \mathbf{h}_{l,T}^{\text{dec}}]^{\text{T}}$ at layer l with the three modules in the layer:

$$\mathbf{H}_l^{\text{dec}} = \text{FFN}(\text{MHA}(\mathbf{Z}_l^{\text{dec}}, \mathbf{R}^{\text{enc}}, \mathbf{R}^{\text{enc}}))\,, \tag{9}$$

$$\mathbf{Z}_l^{\text{dec}} = \text{MHA}(\mathbf{H}_{l-1}^{\text{dec}}, \mathbf{H}_{l-1}^{\text{dec}}, \mathbf{H}_{l-1}^{\text{dec}})\,, \tag{10}$$

$$\mathbf{H}_0^{\text{dec}} = \mathbf{E}^{\text{dec}} + \bar{\mathbf{E}}^{\text{dec}} + \mathbf{R}^{\text{enc}}\,. \tag{11}$$

The intuition of adding the regionalized embeddings $\bar{\mathbf{E}}^{\text{dec}}$ to the decoder inputs is to guide the model to recognize the sequence with the segmentation-level, or chord-level information; and using $\mathbf{R}^{\text{enc}}$ is to take the advantage of the explicit alignment information of the sequence labeling problem for the encoder-decoder architecture [38].

Then, the decoder weighs the hidden states of all layers, as does in the encoder, with the softmax-normalized parameters $\boldsymbol{\alpha}^{\text{dec}} := \{\alpha_l^{\text{dec}}\}_{l=1}^{L}$ to produce the final presentation $\mathbf{R}^{\text{dec}}$ of the decoder inputs:

$$\mathbf{R}^{\text{dec}} = [\mathbf{r}_1^{\text{dec}}, \cdots, \mathbf{r}_T^{\text{dec}}]^{\text{T}} = \sum_{l=1}^{L} \alpha_l^{\text{dec}} \mathbf{H}_l^{\text{dec}}\,. \tag{12}$$

Finally, the representation $\mathbf{R}^{\text{dec}}$ is fed into a fully-connected layer followed by a softmax activation to predict the probability distribution over the chord vocabulary for each time step t in the source sequence:

$$\mathbf{O}^{\text{dec}} = [\mathbf{o}_1^{\text{dec}}, \cdots, \mathbf{o}_T^{\text{dec}}]^{\text{T}} = \text{softmax}(\mathbf{R}^{\text{dec}}\mathbf{W}^{\text{O}}), \tag{13}$$

where $\mathbf{W}^{\text{O}}$ is the parameter matrix of the fully-connected network which maps each $\mathbf{r}_t^{\text{dec}}$ in $\mathbf{R}^{\text{dec}}$ into a vector of the chord vocabulary size. And the *recognition loss* is calculated with $\mathbf{O}^{\text{dec}}$.

We denote the ground truth labels of chord change and chord symbol as $\hat{\mathbf{o}}^{\text{enc}}$ and $\hat{\mathbf{O}}^{\text{dec}}$ respectively. The total loss function L_{total} in the Harmony Transformer is:

$$L_{\text{total}} := \lambda_1 \text{BCE}(\mathbf{p}^{\text{enc}}, \hat{\mathbf{o}}^{\text{enc}}) + \lambda_2 \text{CCE}(\mathbf{O}^{\text{dec}}, \hat{\mathbf{O}}^{\text{dec}})\,, \tag{14}$$

where BCE is the binary crossentropy, CCE is the categorical crossentropy, and λ_1 and λ_2 are coefficients used for balancing the two cross entropies. The two terms in (14) correspond to the segmentation loss and the recognition loss in Figure 1, respectively. [5]

4. EXPERIMENTS

4.1 Data

The proposed model is evaluated on both audio and symbolic datasets. For the audio part, we use the **McGill Billboard** dataset, which consists of 890 musical pieces sampled from the Billboard chart slots [39]. [6] For the symbolic data, we use the **BPS-FH** dataset, in which the first movements of Beethoven's 32 piano sonatas are included [25]. The chord annotations are available in the two datasets; chord segmentation labels are further derived from the chord annotations for the supervised training of the chord segmentation task.

[5] The implementation can be found at `https://github.com/Tsung-Ping/Harmony-Transformer`.

[6] The dataset can be found at `http://ddmal.music.mcgill.ca/research/billboard`.

Dataset	BLSTM	FK	HT	HT*	F1 (Seg.)
Billboard	77.03	78.90	82.68	**83.00**	57.15
BPS-FH	78.87	-	83.96	**84.18**	66.65

Table 1. The chord symbol recognition results in term of WCSR score. BLSTM stands for the 1-layer bidirectional RNN using LSTM cells; both HT and HT* denote the proposed model, except the tonal centroid vector is added into the input feature of the latter. The F1 scores of the chord segmentation of HT are shown in the rightmost column.

Function	BLSTM	HT
Key	72.62	**78.35**
Degree	47.75	**65.06**
Secondary	48.23	**68.15**
Quality	53.31	**74.60**
Inversion	61.59	**62.13**
F1 (Seg.)	-	67.34

Table 2. The accuracy (in %) of harmonic function recognition. Note that *Degree* stands for the accuracy of correctly predicting both the primary and secondary degrees, and *Secondary* indicates the accuracy of correctly predicting the degrees of secondary chords. The segmentation result (with F1 measure) is shown in the bottom row.

4.1.1 Audio data

For each piece in the McGill Billboard dataset, the non-negative-least-squares (NNLS) chromagram is computed with the Chordino VAMP plugin, [40] in which the default settings of the plugin are adopted. Combining the 12-D treble chroma and the 12-D bass chroma, for each track we obtain a 24-by-T-dimensional chromagram, where T represents the length of the track. [7] Each input sequence for the Harmony Transformer contains 100 segments (around 23 sec), and is generated through a sliding window of frame size 21 with hop size 5. Following [13] (see Section 2.2), pieces with id numbers smaller than 1000 are used for training, and the remaining for testing; also, identical pieces are filtered out. As a result, there are 5,647 sequences for training and 1,628 sequences for testing.

4.1.2 Symbolic data

We represent the piano sonatas in the BPS-FH dataset as pianorolls with the pitch ranging from A0 to G$^\#$7 (middle C = C4), where the duration of each note is measured in term of crotchet beats, and is quantized to 32th note. The length of each sequence is 64, and each element in the sequence is a pianoroll segmented with window size of 33 and hop size of 2. As the input element for the Harmony Transformer, each pianoroll segment is flattened into a vector whose length is 84×33. We use 4-fold cross-validation for evaluation; the number of sequences of each fold varies from 368 to 585.

4.1.3 Data augmentation

All the training data are augmented with key modulation and thus expanded to 12 times.

4.2 Experimental trials

For the Harmony Transformer (denoted as HT), we use the embedding size $d = 512$, the number of heads $h = 8$, and the number of layers of both the encoder and the decoder $L = 2$. The two coefficients λ_1 and λ_2 in the loss function are set to be 3 and 1 respectively. The chord symbol recognition task is conducted for both audio and symbolic data, and the harmonic function recognition task, as defined in [25], is further applied to the symbolic dataset. Besides, we employ the *tonal centroid* vector [20], which models the relationship between chords in a 6-D tonal space, as the additional input feature of the HT for better representing the input data.

4.2.1 Chord symbol recognition

The chord symbol recognition model has a 26-dimensional output, in which 24 of them represent major and minor triads, 1 represents 'others' for chords other than major or minor triads, and the remaining one represents the 'no-chord' case. The weighted chord symbol recall (WCSR) is used as the evaluation metric. We employ a 1-layer bidirectional RNN using LSTM cells of 512 hidden units (abbreviated as BLSTM) as the baseline for the evaluations of the two datasets. Additionally, we include the best evaluation result achieved by the ConvNet-HMM model (denoted as FK here) in [13] (see Section 3) for the comparison of the evaluation on audio data. The FK model is similar to the Madmom audio chord recognition framework, [8] which achieves many state-of-the-art scores in the MIREX Audio Chord Estimation (ACE) campaign. [9]

4.2.2 Harmonic function recognition

We formulate the harmonic function recognition task as a multi-task learning problem, in which the model outputs segment-wise predictions of the 5 chord functions: *local key* of 24 classes, *primary degree* of 21 classes, *secondary degree* of 21 classes, *chord quality* of 10 classes, and *chord inversion* of 4 classes. We use the classification accuracy to measure the performance of the proposed model. The BLSTM, as mentioned in Section 4.2.1, is also employed for comparison. For more information about the terminology of harmonic function, please refer to [25].

[7] For more information of the Chordino VAMP plugin and the NNLS chromagram, please refer to `http://www.isophonics.net/nnls-chroma`.

[8] `https://madmom.readthedocs.io/en/latest/modules/features/chords.html`.

[9] See `https://www.music-ir.org/mirex/wiki/2018:Audio_Chord_Estimation_Results`. The algorithm denoted as FK2 in the website is part of the Madmom audio processing framework.

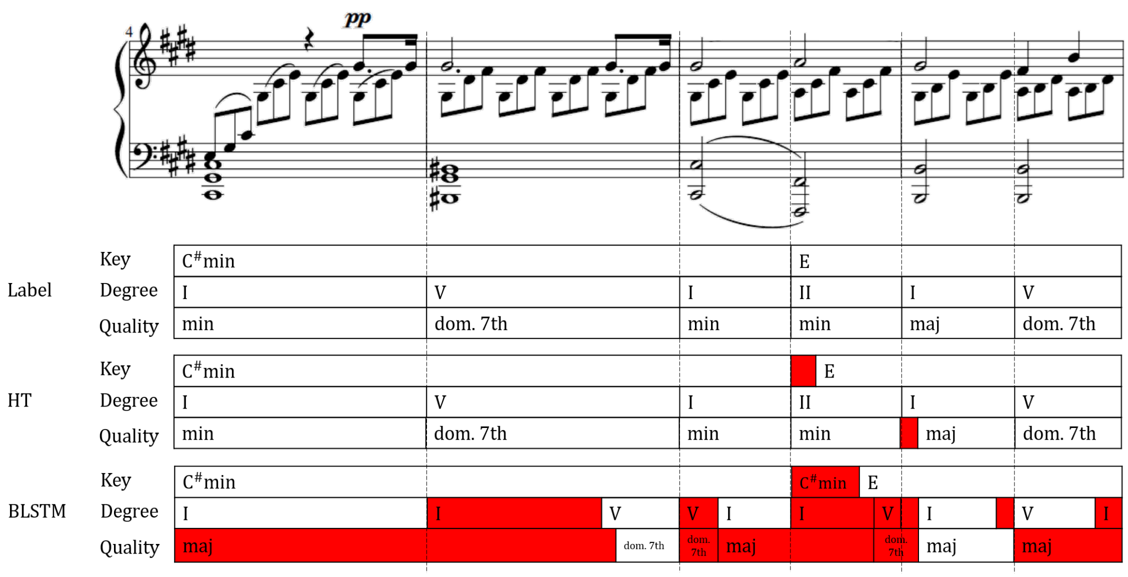

Figure 2. The predictions of harmonic function by the proposed model HT and the BLSTM, where *maj*, *min*, and *dom. 7th* stand for major, minor, and dominant seventh, respectively. Wrong predictions are marked in red. The example here is an excerpt from Beethoven's piano sonata No. 14, Op. 27-2, Mvt. 1, MM. 4-7.

4.3 Hyperparameters and training

The model is trained with the Adam optimization method of the learning rate $= 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. To avoid overfitting, we employ dropout [41] of rate $= 0.6$ and label smoothing [42] of the value $\epsilon_{ls} = 0.1$. In addition, because the encoder output contains discrete variables, i.e., the binary chord segmentation predictions $\mathbf{o}^{\text{enc}}$, we utilize a straight-through estimator along with the slope annealing trick [43] to estimate the derivative gradient during backpropagation.

4.4 Evaluation results

Table 1 shows the evaluation results of the chord symbol recognition task. For the Billboard dataset, the performance of the Harmony Transformer outperforms the BLSTM and the FK by 5.65% and 3.78% respectively. For the BPS-FH dataset, our model also surpasses the BLSTM by 5.09%. When we employ the tonal centroid vector for the input feature, the proposed model further boosts the recognition results. This indicates that the explicit information of harmonic change is useful for chord recognition. The fact that the proposed model outperforms the BLSTM in both audio and symbolic data exhibits our model's capability of sequence learning even though the model consists of no temporally recurrent computation. In addition, based on the same training and evaluation data, the proposed model performs better than the FK and therefore may compete with the ACE framework of Madmom. Nevertheless, the F1 scores for the segmentation task are not satisfied due to the low recall rates (57.10% and 58.69% for Billboard and BPS-FH). This indicates the challenge to identify the exact time when chord changes.

For the harmonic function recognition task, the HT outperforms the BLSTM in all of the five chord functions, as shown in Table 2. In particular, for local key, chord degree, secondary chord degree, and chord quality, the HT outperforms the BLSTM greatly by 5.73%, 17.31%, 19.92%, and 21.29%, respectively. This indicates that our model is able to deal with challenging cases such as the key modulation, and chords with special harmonic function. Figure 2 gives an example of such instances. The HT correctly predicts the chord progression in terms of local key, chord degree, and chord quality, except that the timing of modulating to E major is slightly later than the ground truth. Notably, the F minor triad at the second half of measure 6, functioned as a pivot chord in a common chord modulation, is precisely identified by our model in spite of the key change. In contrast, the BLSTM not only fails to recognize the degree of some chords in this example, but also mistakes the minor triads as major ones, such as in measure 4.

5. CONCLUSION AND FUTURE WORK

We have demonstrated that the Harmony Transformer is competent in harmony analysis. Built upon the commonality of musical chord recognition and natural language processing, we emphasize the importance of modeling segmental and hierarchical structures in music. The encoder-decoder architecture and the non-autoregressive decoding in the Harmony Transformer enhance the flexibility in modeling chord sequences. Specifically, the end-to-end combination of chord segmentation and chord recognition contributes great benefits to the chord symbol recognition, as well as to the joint recognition of five harmony functions, a challenging task that relies heavily on the contextual and structural information in music. Our model has the potential to be further improved by using the segmentation results to learn the *word-level* embedding, which has also witnessed success in natural language processing.

6. REFERENCES

[1] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 37–43, 2016.

[2] Eric J. Humphrey and Juan P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, page 357–362, 2012.

[3] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR)*, pages 52–58, 2015.

[4] Eric J. Humphrey and Juan P. Bello. Four timely insights on automatic chord estimation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 673–679, 2015.

[5] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

[6] Tristan Carsault, Jérôme Nika, and Philippe Esling. Using musical relationships between chord labels in automatic chord extraction tasks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 18–25, 2018.

[7] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 335–340, 2013.

[8] Junqi Deng and Yu-Kwong Kwok. Large vocabulary automatic chord estimation with an even chance training scheme. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 531–536, 2017.

[9] Takeshi Hori, Kazuyuki Nakamura, and Shigeki Sagayama. Music chord recognition from audio data using bidirectional encoder-decoder LSTMs. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1312–1315, 2017.

[10] Filip Korzeniowski, David R. W. Sears, and Gerhard Widmer. A large-scale study of language models for chord prediction. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 91–95, 2018.

[11] David R. W. Sears, Filip Korzeniowski, and Gerhard Widmer. Evaluating language models of tonal harmony. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 211–217, 2018.

[12] Brian McFee and Juan P. Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 188–194, 2017.

[13] Filip Korzeniowski and Gerhard Widmer. On the futility of learning complex frame-level language models for chord recognition. In *Proceedings of the Audio Engineering Society (AES) International Conference on Semantic Audio*, 2017.

[14] Filip Korzeniowski and Gerhard Widmer. Improved chord recognition by combining duration and harmonic language models. In *Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR)*, pages 10–17, 2018.

[15] Yiming Wu and Wei Li. Music chord recognition based on midi-trained deep feature and BLSTM-CRF hybird decoding. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 376–380, 2018.

[16] W. Bas de Haas, José Pedro Magalhães, and Frans Wiering. Improving audio chord transcription by exploiting harmonic and metric knowledge. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 295–300, 2012.

[17] Veronika Zenz and Andreas Rauber. Automatic chord detection incorporating beat and key detection. In *Proceedings of the IEEE International Conference on Signal Processing and Communications (ICSPC)*, pages 1175–1178, 2007.

[18] Kyogu Lee and Malcolm Slaney. Automatic chord recognition from audio using an HMM with supervised learning. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, pages 133–137, 2006.

[19] Ruofeng Chen, Weibin Shen, Ajay Srinivasamurthy, and Parag Chordia. Chord recognition using duration-explicit hidden markov models. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 445–450, 2012.

[20] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26, 2006.

[21] Matthias Mauch, Katy C. Noland, and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International*

Society for Music Information Retrieval Conference (ISMIR), pages 231–236, 2009.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, pages 4171–4186, 2019.

[24] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *arXiv preprint arXiv: 1810.12247*, 2018.

[25] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 90–97, 2018.

[26] Hakan Erdogan. Sequence labeling: generative and discriminative approaches, hidden markov models, conditional random fields and structured SVMs. In *the Tutorial of International Conference on Machine Learning and Applications (ICMLA)*, 2010.

[27] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations (ICLR)*, 2018.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *arXiv preprint arXiv: 1607.06450*, 2016.

[30] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237, 2018.

[31] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.

[32] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R. Glass. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 861–872, 2017.

[33] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1923–1933, 2017.

[34] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. In *arXiv preprint arXiv: 1308.3432*, 2013.

[35] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional generative adversarial networks with binary neurons for polyphonic music generation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 190–196, 2018.

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, pages 1923–1933, 2015.

[37] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1243–1252, 2017.

[38] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association (ISCA)*, pages 685–689, 2016.

[39] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 633–638, 2011.

[40] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 135–140, 2010.

[41] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, pages 1929–1958, 2014.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[43] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.