creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

**저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.**

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학박사학위논문

# Automatic Music Lead Sheet Transcription and Melody Similarity Assessment Using Deep Neural Networks

## 심층 신경망 기반의 음악 리드 시트 자동 채보 및 멜로디 유사도 평가

2023 년 2 월

서울대학교 대학원

산업공학과

박 종 권

# Automatic Music Lead Sheet Transcription and Melody Similarity Assessment Using Deep Neural Networks

## 심층 신경망 기반의 음악 리드 시트 자동 채보 및 멜로디 유사도 평가

지도교수 이 경 식

이 논문을 공학박사 학위논문으로 제출함

2022 년 12 월


서울대학교 대학원

산업공학과

박 종 권


박종권의 공학박사 학위논문을 인준함

2022 년 12 월

| | | |
|---|---|---|
| 위 원 장 | 조 성 준 | (인) |
| 부위원장 | 이 경 식 | (인) |
| 위 원 | 박 종 헌 | (인) |
| 위 원 | 이 재 욱 | (인) |
| 위 원 | 박 종 혁 | (인) |

# Abstract

# Automatic Music Lead Sheet Transcription and Melody Similarity Assessment Using Deep Neural Networks

Jonggwon Park
Department of Industrial Engineering
The Graduate School
Seoul National University

Since the composition, arrangement, and distribution of music became convenient thanks to the digitization of the music industry, the number of newly supplied music recordings is increasing. Recently, due to platform environments being established whereby anyone can become a creator, user-created music such as their songs, cover songs, and remixes is being distributed through YouTube and TikTok. With such a large volume of musical recordings, the demand to transcribe music into sheet music has always existed for musicians. However, it requires musical knowledge and is time-consuming.

This thesis studies automatic lead sheet transcription using deep neural networks. The development of transcription artificial intelligence (AI) can greatly reduce the time and cost for people in the music industry to find or transcribe sheet music. In addition, since the conversion from music sources to the form of digital music

is possible, the applications could be expanded, such as music plagiarism detection and music composition AI.

The thesis first proposes a model recognizing chords from audio signals. Chord recognition is an important task in music information retrieval since chords are highly abstract and descriptive features of music. We utilize a self-attention mechanism for chord recognition to focus on certain regions of chords. Through an attention map analysis, we visualize how attention is performed. It turns out that the model is able to divide segments of chords by utilizing the adaptive receptive field of the attention mechanism.

This thesis proposes a note-level singing melody transcription model using sequence-to-sequence transformers. Overlapping decoding is introduced to solve the problem of the context between segments being broken. Applying pitch augmentation and adding a noisy dataset with data cleansing turns out to be effective in preventing overfitting and generalizing the model performance. Ablation studies demonstrate the effects of the proposed techniques in note-level singing melody transcription, both quantitatively and qualitatively. The proposed model outperforms other models in note-level singing melody transcription performance for all the metrics considered. Finally, subjective human evaluation demonstrates that the results of the proposed models are perceived as more accurate than the results of a previous study.

Utilizing the above research results, we introduce the entire process of an automatic music lead sheet transcription. By combining various music information recognized from audio signals, we show that it is possible to transcribe lead sheets that express the core of popular music. Furthermore, we compare the results with lead sheets transcribed by musicians.

Finally, we propose a melody similarity assessment method based on self-supervised learning by applying the automatic lead sheet transcription. We present convolutional neural networks that express the melody of lead sheet transcription results in embedding space. To apply self-supervised learning, we introduce methods of generating training data by musical data augmentation techniques. Furthermore, a loss function is presented to utilize the training data. Experimental results demonstrate that the proposed model is able to detect similar melodies of popular music from plagiarism and cover song cases.

**Keywords**: Music Information Retrieval, Automatic Music Transcription, Chord Recognition, Singing Melody Transcription, Melody Similarity Assessment, Music Plagiarism Detection, Self-supervised Learning, Deep Neural Networks

**Student Number**: 2018-20381

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Music is the most familiar cultural content to the public. Thanks to the recent development of the streaming industry, anyone can listen to whatever music they want anywhere. Since the composition, arrangement, and distribution of music has become convenient thanks to the music industry's digitalization, the number of newly made music recordings is continually increasing. Recently, thanks to the growth of video content, music has become more frequently present on platforms such as YouTube and TikTok. In addition, since everyone can become a creator, user-created music such as their own songs, covers, and remixes are being distributed through YouTube and SoundCloud.

The demand to transcribe music as diverse versions of newly released music has always existed people in the music industry and hobbyist players. However, not anyone can transcribe music scores because it requires musical knowledge and senses; it also has issues such as time and financial expenses. Since new music contents keep flowing, such as new songs and cover songs, it is gradually becoming impossible for people to write scores for all music content. In addition, recent issues have arisen regarding music plagiarism. Analyses of plagiarism, rather than relying simply on

the similarity of sounds, requires analysis of music transcriptions for the evaluation of similarity on fundamental chords and melodies.

Meanwhile, alongside the adoption of deep learning technology, thanks to the development of various techniques such as transformers [2,3] that use sequence data, big developments have been made in diverse sectors such as translation [2], language generation [4], and voice synthesis [5]. As deep learning is used to apply music audio signals, which is a type of sequence data, there have been noteworthy achievements in various sectors, such as source separation [6], music audio synthesis [7], and music similarity measurement [8].

Various attempts have been made in the field of automatic music transcription (AMT). Attempts to transcribe chords, which are a key element of music, have been made over a long period [9]. Deep learning has enabled many recent developments [10, 11]. In addition, traditional fundamental frequency analysis [1] has been frequently employed to recognize melodies. Recently, melody recognition at the note level has been attempted [12]. Further, studies have been conducted to transcribe various musical instruments, such as pianos, guitars, and drums [13]. An audio-to-score study [14] has been conducted to convert music audio into sheet music, and a study on lead sheet transcription [15] is in progress.

There is no general rule defining plagiarism, which states that at least a few notes or beats must exist simultaneously in music to be considered to be infringing on music copyright [16]. When courts deal with music plagiarism cases, independent music experts evaluate the similarities between the two songs to make a determination [17]. There are many kinds of music plagiarism, but these can largely be divided into sample plagiarism, melody plagiarism, and rhythm plagiarism [17]. Among these, melody

is one of the most important elements of popular music, which makes it the most studied area of plagiarism research [18–20]. There have been studies on the exploration of plagiarism on audio signals [17, 21], but it is difficult to apply to actual cases of popular music plagiarism, which involve various musical instruments and modifications.

This thesis started with the motivation that if a high level of music transcription is possible through deep learning technology, various demands will be satisfied and additional development of the application will be possible. The study will provide assistance to those who lack musical knowledge with an analysis of music while it will reduce the time and cost for experts who have the ability to transcribe. Furthermore, the analysis of music core elements using the music transcription technology will enable applications such as the development of musical elements' similarity-based search and recommendation system.

## 1.2 Objectives

This thesis aims to propose a lead sheet transcription technique using deep neural network-based analysis of music audio signals. Lead sheet is a musical notation that specifies the essential elements of a popular song; it consists of musical key, chords, and melody. To express audio signals in the form of a lead sheet, we suggest chord recognition and singing melody transcription models. Furthermore, based on the result, this thesis defines the post-processing procedure of converting it into a lead sheet. In addition, by applying the lead sheet transcription result, this thesis proposes an melody similarity assessment technique for automatic music plagiarism detection and cover song detection. This study is divided into four parts as follows.

First, we study the chord recognition technique from music audio signals. Chords refer to a set of two or more pitches, and alongside the melody, they are a basic element of music composition. Chord recognition is difficult for several reasons; not all notes of the current chords are always played simultaneously and there are many chords with similar meanings. Furthermore, it can be difficult to determine the point at which the chord changes. Therefore, for the analysis of music audio signals, the study uses the attention technique that creates representation values for each point. This study proposes the attention-based model and learning techniques to improve performance.[1] In addition, the study visualizes and analyzes the attention values of the chord division method of the chord recognition model.

Second, the thesis proposes a note-level singing melody transcription Transformer to recognize the monophonic singing melody from polyphonic audio signals.[2] Mono-

[1]The work in Chapter 4 was published as Park et al. [22].
[2]The work in Chapter 5 was published as Park et al. [23].

phonic note event tokens are defined to express a monophonic melody as a sequence of event tokens. Furthermore, we propose three techniques to enhance the transcription performance, namely overlapping decoding to resolve a context breakage between segments in decoding, pitch augmentation to enlarge the training dataset, and adding noisy dataset with data cleansing. The experimental results imply that combining the proposed methods significantly improves the performance. The proposed model outperforms other models in note-level singing melody transcription regarding note-level evaluation metrics. Through the analysis of F0 estimation evaluation metrics, we show that the voice detection performance of the proposed model is comparable to that of a previous study. Finally, the visualization of the results and subjective listening test demonstrate that the proposed methods are effective in achieving better transcriptions.

Third, using the recognized chords and melody, this thesis suggests methods for conversion into the form of a lead sheet. To express music in the form of lead sheet, aside from chords and melody, musical key information and beat tracking are required for conversion into note units. For this, the study utilizes previously published research results. As a result, the study identifies that it is possible to transcribe lead sheets from audio signals.

Finally, the thesis suggests melody similarity assessment method by applying lead sheet transcription. This study proposes methods to evaluate similarity based on recognized melodies. For this, the study adopts the self-supervised learning technique to learn the similarities between unlabeled melodies. Through data augmentation using musical theory, the study utilizes the data by transforming it adequately. Further, by applying this into actual plagiarism detection and cover song identification,

the study examines the detection performance of proposed models.

## 1.3 Thesis Outline

The thesis comprises eight chapters, and the remaining chapters are organized as follows. Chapter 2 examines the previous research on automatic music transcription models and music plagiarism detection. Chapter 3 describes the problem definition used in the thesis. In Chapter 4, a bi-directional Transformer for musical chord recognition is proposed. Chapter 5 introduces note-level singing melody transcription method with Transformers. Combining the results of Chapters 4 and 5, automatic music lead sheet transcription method is proposed in Chapter 6. Applying the results of Chapter 6, melody similarity assessment with self-supervised covolutional neural network is suggested in Chapter 7. Finally, the conclusion of this thesis and future research directions are presented in Chapter 8.

# Chapter 2

# Literature Review

## 2.1 Attention Mechanism and Transformers

### 2.1.1 Attention-based Models

The attention mechanism, first introduced by [24], can be described as computing an output vector when query, key and value vectors are given. In sequence modelling tasks such as machine translation, query and key correspond to certain elements of the target sequence and the source sequence respectively. Each key has its own value. The output is computed as a weighted sum of the values where the weights are computed from the query and key. Self-attention refers to the case when query, key and value are computed from the same input.

As depicted in Figure 2.1, Transformer [2] is an attention-based network that relies on attention mechanism only and does not include recurrent or convolutional architecture. Utilizing multi-head attention together with position-wise fully-connected feed-forward network, it showed significantly faster training speed and achieved better performance than recurrent or convolutional networks for translation tasks.

Transformer used scaled dot-product as an attention function:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}})V \tag{2.1}$$

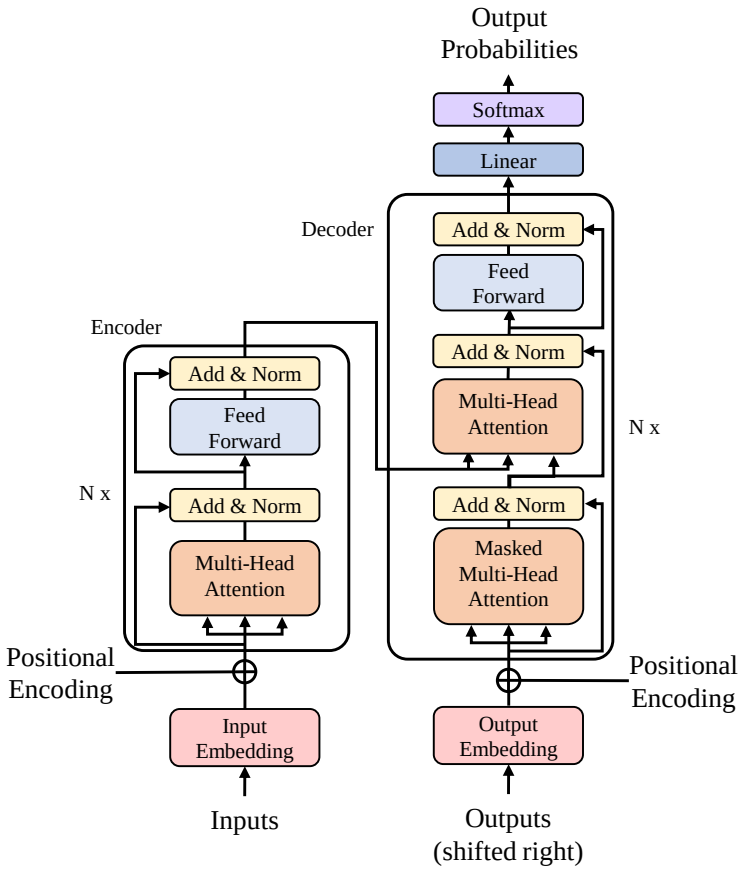



Figure 2.1: Model architecture of Transformer [2], a self-attention-based sequenceto-sequence model

where $Q$, $K$ and $V$ are matrices of query, key and value vectors respectively, and $d_K$ is the dimension of key.

### 2.1.2 Transformers with Musical Event Sequence

Transformer [2], which demonstrated outstanding performance in various tasks dealing with sequence data, was also often used in the field of MIR. Music Transformer [25] generates piano music as a sequence of note onset, note offset, set velocity, and time-shift event tokens. The time-shift tokens express relative time, indicating

that the next event occurs after a certain amount of time follwing the preceding event. Choi et al. [26] investigated chord conditioned melody generation by symbolizing a monophonic melody as a sequence of pitch onset, hold, and rest tokens, where each token in sequence is defined to be the length of a 16th-note.

The sequence-to-sequence Transformer has achieved state-of-the-art performance in the task of piano transcription [27]. Despite the different lengths of audio feature and musical event token sequences, the authors proposed to use audio features as the encoder input and train the decoder to predict the output tokens autoregressively. A polyphonic piano performance was represented as a sequence of tokens where the token set consists of absolute time, note, velocity, and end of sequence (EOS). The results of the experiment comparing relative time-shift and absolute time tokens demonstrated that the latter performed better in piano transcription, due to the prevention of error accumulation. Following the results, we also adopted absolute time tokens for data representation.

In [13], a multi-task AMT Transformer was proposed, confirming that transcription of arbitrary combinations of instruments is possible by training various note-level instrument datasets simultaneously. The musical event sequence used in the decoder consists of instrument, note, on/off, time, drum, end tie section, and EOS. The instrument tokens enable distinguishing notes of different instruments. Moreover, the authors introduced an end tie section token as a method of conveying information about notes that were not turned off in the previous segment. The multi-task AMT Transformer obtained high-quality transcription results on various instrument datasets, but the task of singing transcription was not covered.

In this paper, referring to [27] and [13], a sequence-to-sequence Transformer is

applied to note-level singing transcription. While [27] focused on polyphonic piano transcription, this paper approaches the problem of monophonic singing melody transcription.

## 2.2 Chord Recognition

In the past, most automatic chord recognition systems were divided into three parts: feature extraction, pattern matching and chord sequence decoding. After applying transformation such as short-time Fourier transform or constant-q transform (CQT) to an input audio signal, features are extracted from the resulting time-frequency domain. Some examples of such hand-crafted features include chroma vectors and the "Tonnetz" [28] representation. For pattern matching and chord sequence decoding, Gaussian mixture models with feature smoothing [29, 30] and HMMs [9, 31] have been the most popular choices, respectively.

With the recent wide acceptance of deep learning in research communities, there have been many studies applying it to chord recognition task in various ways. The very first deep-learning-based chord recognition system was proposed by [32] where they trained a CNN for major-minor chord classification. Attempts to apply deep learning to feature extraction include [33] and [34], where the former employed a CNN to extract Tonnetz features from audio data and the latter adopted a deep neural network (DNN) to compute chroma features. CNN and HMM were combined for chord recognition in [35] and [36].

In addition to CNN, another popular network architecture for chord recognition is RNN. [37] and [38] explored an RNN as chord sequence decoding method, relying on deep belief network and a DNN, respectively. Another branch of RNN-based chord recognition systems utilize a language model which predicts only the sequence of chords without considering their durations. This might be helpful when the number of chord labels is large. A large-scale study of language models for chord prediction was conducted in [39]. Without audio data, the authors trained just a language

model with the chord progression data only and showed that RNNs outperformed N-gram models. In their succeeding work [10], they combined the RNN-based harmonic language model with a chord duration model to complete the chord recognition task.

Another RNN-based approach is presented in [11] which trained a CNN feature extractor with large MIDI (Musical Instrument Digital Interface) data and combined BLSTM (Bi-directional Long-Short Term Memory) with CRF for sequence decoder. This BLSTM-CRF model achieved good performance but has a drawback that its training procedure involves complex MIDI pre-training. The model that we propose, on the other hand, is much simpler to train.

## 2.3 Note-level Singing Melody Transcription

For note-level singing transcription, it is essential to recognize the pitch, onset, and offset of each note. Tony [40] enabled interactive annotation of melodies from monophonic audio recordings. It supported note-level transcription by first performing pitch tracking using pYIN [41] and then converting the F0 results into note-level annotation with hidden Markov model [42]. Furthermore, consecutive notes of similar pitch were segmented by applying the amplitude-based onset segmentation heuristic.

Omnizart [43] provided various AMT functions based on deep learning, such as vocal transcription, chord recognition, drum transcription, and beat tracking. Its vocal transcription module for polyphonic music adopted a hybrid network comprising frame-level pitch extraction and note segmentation models. The authors used pretrained Patch-CNN [44] for pitch tracking and improved the previously proposed note segmentation model using Pyramid-Net with ShakeDrop regularization [45] and virtual adversarial training [46].

The downside of various note-level singing transcription datasets is that the amount of data is small or the annotations are inaccurate. Wang et al. [12] proposed a large-scale dataset for singing transcription consisting of 500 pop songs (MIR-ST500) by setting some labeling criteria and obtaining annotations from non-experts. With the proposed dataset, they proposed a singing transcription model that recognizes onset, silence, pitch, and octave for each time frame using EfficientNet-b0. EfficientNet-b0 [47] is a convolutional neural network model that showed state-of-the-art performance on image classification , while keeping the model size small compared to other models. The proposed model in this work was also trained and evaluated with MIR-ST500. Furthermore, although Wang et al. [12] stated that DALI

has inevitable errors, we make use of the dataset as an additional noisy dataset.

Kum et al. [48] proposed a semi-supervised learning method to solve the problem of insufficient note-level labeled data in singing transcription from polyphonic music. The authors generated pseudo-labels by applying pitch and rhythm quantizations to the results of a vocal pitch estimation model. The proposed singing transcription model was trained with unlabeled audio data and the pseudo-labels. Furthermore, the repeatedly applied self-training using the teacher-student framework [49] led to additional performance improvement. They showed that the use of unlabeled data in addition to labeled data can improve the performance of the singing transcription model.

Donahue et al. [15] designed a system that produces lead sheets from music audio. The authors claimed that using the audio feature of Jukebox [7] as input instead of spectrogram features led to significant performance improvement when training a Transformer [2] for melody transcription. However, only the performance for note onset was reported; the specific training technique was not disclosed. This paper proposes a note-level singing transcription Transformer and analyzes its performance using various metrics.

## 2.4 Musical Key Estimation

The musical key is a group of pitches that form the basis of music. The most general key categorization method is expressing it as one of 24 types by dividing it into 12 pitches, major and minor. Studies to recognize the musical key from audio signals have largely been conducted in the field of music information retrieval with the development of digital signal processing technology. The early-stage key prediction models [50–52] follow similar methods. First, audio signals are represented in the form of time-frequency, which is converted into a chroma vector. Next, the resultant chroma vector is compared with template vector for each key to predict to which key it is the most similar. Such methods rely on hand-crafted features and therefore have limitations such as low generalization performance.

Thanks to the introduction of deep learning on audio signal processing, many studies have investigated key estimation. For example, [53] utilized convolutional neural network to propose an end-to-end system for musical key estimation; this data-driven model showed improved performance compared to relying on existing hand-crafted features. Meanwhile, [54] proposed a model to overcome the issue of large disparity of performance of key estimation models depending on the genre. Likewise, convolutional neural network was used to propose the key estimation model that is genre-agnostic through changes of model structures and a learning process. There has also been research analyzing the impact on the key estimation model depending on the convolutional neural network's filter type [55]. In the case of key, spectral characteristics—as opposed to temporal—were more important and it is effective to use a suitable filter for this.

Recently, since chords and keys share musical characteristics, methods have been

suggested that can predict them simultaneously through multi-task learning [56]. Aside from the classification model, a method to give a regularization effect by utilizing variational auto-encoder and language model structure together has been suggested. Multi-task learning for chord and key prediction has improved the performance of key estimation.

In this thesis, key information is predicted using [54] for automatic lead sheet transcription.

## 2.5 Beat Tracking

In music, the beat is a feature that represents the rhythmical characteristics. Beat tracking is a matter of guessing when each beat exists in an audio signal. Among beats, a strong beat at the beginning of a measure is called a downbeat. Since a downbeat can distinguish each bar, it can be used to analyze a song's structure.

In the early stages, the following methods were often used for beat tracking [57]. Features are extracted from audio signals and the periodicity—represented as tempo—within this is discovered. And through phase detection, the position of the beat was estimated. These methods mostly required feature engineering and post-processing.

Thanks to the introduction of deep learning, many studies have attempted data-driven methods of learning the beat tracking model. At [58], bi-directional Long Short-term Memory [59], which is a recurrent neural network, was applied to beat tracking. Instead of the existing complicated feature engineering work, audio signals' spectral features were used and the beat activation function was created directly through the neural network. Further, the position of each beat could be predicted through peak detection. In [60], researchers suggested models to predict the beat and downbeat together as they are both directly involved. Applying the recurrent neural network directly into the magnitude spectrogram provided the output features. Then, using a dynamic Bayesian network, the method to predict the beat and downbeat positions could be suggested. the study identified good performance in diverse genre and styles. Meanwhile, [61] suggested a multi-task method to predict the tempo and beat that are closely related. Predicting tempo and beat at once had the effect of improving the performances of tempo estimation and beat tracking.

In [62], a temporal convolutional network beyond the existing recurrent approach was suggested; the performance was best on existing beat-tracking datasets and the training and computation was efficient.

For real-time beat tracking, [63] suggested methods of using predominant local pulse information. Meanwhile, [64] suggested using a recurrent neural network and enhanced particle filtering to perform online beat tracking method. This suggested method improved the performance of online beat tracking, thereby showing a similar level of results as offline methods.

This thesis used the model studied in [60] as it requires beat information to represent melodies and chords as a lead sheet.

## 2.6 Music Plagiarism Detection and Cover Song Identification

Music plagiarism detection techniques are largely divided into audio and symbolic based methods. In Dittmar et al. [17], since automatic music transcription technology has not been sufficiently developed, one limitation was proposing an idea of monitoring melody plagiarism using pitch vector similarity and sequence alignment. Sie et al. [21] proposed a method in which the fundamental frequency was extracted from audio signals to find plagiarized parts using path finding. However, this was heavily reliant on the fundamental frequency recognition performance and is difficult to use, other than for songs with only human voices or humming, such as in general popular music, since their melodies are typically polyphonic. Borkar et al. [65] proposed a music plagiarism detection method using audio fingerprinting and sequence matching technology. However, there is an issue in the characteristics of audio fingerprinting, namely that the performance becomes greatly reduced when facing even a small transformation. It is appropriate for sample plagiarism detection but difficult to operate for detection on melodies when sung by a completely different singer.

Prisco et al. [18] proposed a music plagiarism detection method based on fuzzy vector similarity by expressing symbolic melody rhythm and pitch as vectors. He et al. [19] introduced a music plagiarism detection technique that could deal with issues of shift, transposition, and tempo-variance problems using of bipartite graph matching. Through this, locally similar sections can be found even among two songs that have overall low similarity. Park et al. [20] represented a symbolic melody as an image of piano roll forms and suggested a Siamese CNN-based plagiarism

detection technique. However, symbolic melody-based detection techniques have low applicability due to the limitation that they cannot be used for music audio signals.

The cover song identification task is a matter of identifying the same song sung by different people. It is not the same as general plagiarism detection but can be seen as similar in that it finds cases where a similar melody is newly recorded. [66] approached the cover song identification matter using convolutional neural network. First, the model is trained through classification using labels and then the model is used to extract the representation from audio signals, which was used in the cover song identification. [67] also studied cover song identification by partly transforming the ResNet [68] structure. It suggested a structure to learn triplet loss and classification loss simultaneously. Through this, it was learned to extract the invariant feature on key, tempo, timbre, genre while preserving the information on the type of the song. Arcos [69] suggested the method of finger printing based on chord and melody for cover song detection of western classical music. This method was used for cover song detection by utilizing the chord and melody results extracted from the audio signal as features.

This thesis deals with singing melody plagiarism detection and cover song detection through melody similarity assessment.

## 2.7 Deep Metric Learning and Triplet Loss

Deep metric learning maps input into feature vectors using the deep neural networks. The distance between the feature vectors of different inputs in this manifold space can thereby be calculated. Deep neural network, which maps the input into feature vector is usually learned through stochastic gradient descent, for which the loss function needs to be defined. It should be learned that the feature vectors of the same label's data need to be close, while the feature vectors of different label's data need to be far away in manifold space.

The most frequently used loss function of deep metric learning is triplet loss [70], which calculates the loss using a pair of dataset consist of anchor, positive sample, and negative sample. An anchor is the standard input data, while a positive sample is data that is either same or similar to the anchor. A negative sample is data not relevant to the anchor. Therefore, it should be represented on the manifold that the anchor and positive sample are close while anchor and negative sample is far. This can be expressed visually, as in Figure 2.2.

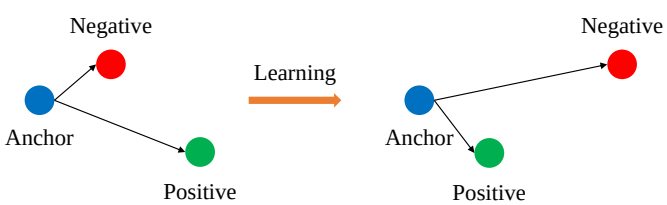



Figure 2.2: Minimizing triplet loss makes the distance between anchor and positive sample (which have the same label) close while making the distance between anchor and negative sample far, which have different labels.

The triplet loss function is as shown in Equation 2.2.

$$Triplet(a, p, n, \alpha) = \sum_{i}^{N} \left[ \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{2.2}$$

$a$, $p$ and $n$ are anchor, positive sample, and negative sample, respectively, while $f$ is embedding function consisting of deep neural network. $\alpha$ is the margin, and $N$ refers to the total number of data pairs. Deep neural network is learned to minimize this loss.

This thesis uses triplet loss to represent the similar melodies as close in manifold space for music plagiarism detection.

# Chapter 3

# Problem Definition

## 3.1 Lead Sheet Transcription

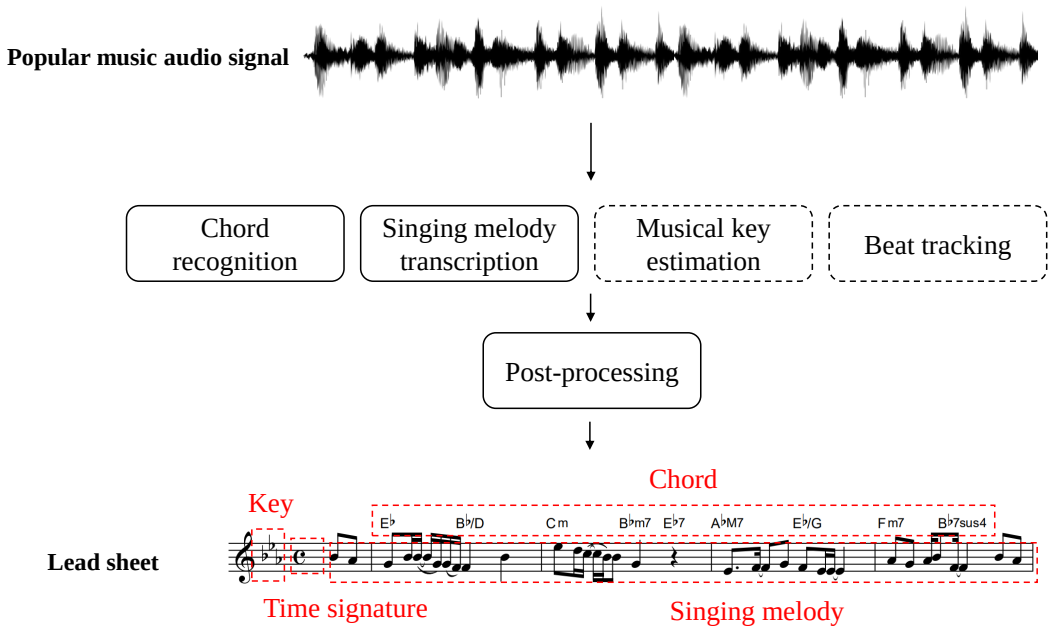



Figure 3.1: The whole process of automatic lead sheet transcription.

Popular songs consist of diverse musical instruments, sound, and singing. A lead sheet is a form of notes to briefly indicate melody, lyrics, and chords, which are the essential elements of popular songs. Lead sheet transcription is a matter of converting the popular song's music audio signals into a lead sheet. The entire process can be

expressed visually, as in Figure 3.1. For this, chord, singing melody, key, and beat need to be recognized and the results need to be combined to be finally converted into lead sheet through post-processing. This thesis does not include lyric recognition. Existing research outcomes are used for key and beat recognition while this thesis focuses on chord and melody recognition.

### 3.1.1 Chord Recognition

Chords are highly abstract and descriptive features of music that can be used for a variety of musical purposes, including automatic lead-sheet creation for musicians, cover song identification, key classification and music structure analysis [71–73]. Since manual chord annotation is labor intensive, time consuming and requires expert knowledge, automatic chord recognition system has been an active research area within the music information retrieval community. Automatic chord recognition is challenging due to the fact that 1) not all the notes played are necessarily related to the chord of the moment and 2) simple one-hot encoding of chord labels cannot represent the inherent relationship between different chords.

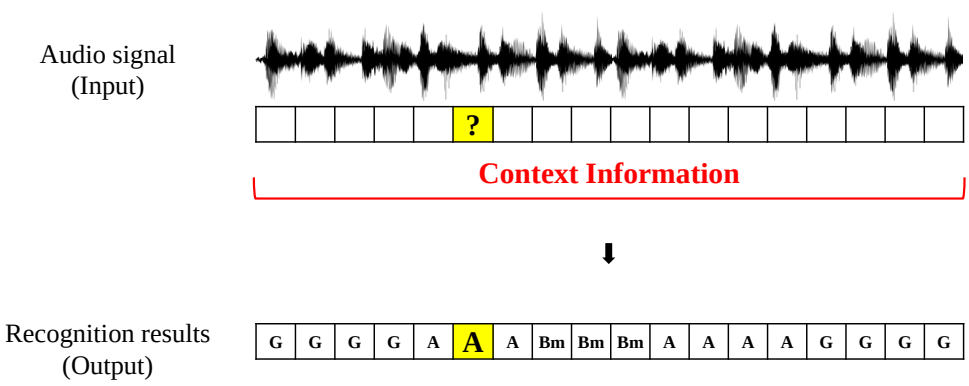



Figure 3.2: Problem definition of chord recognition.

The goal of chord recognition task is to output a sequence of time-synchronized chord labels when a raw audio recording of music is given as input. The chord recognition can be expressed visually, as shown in Figure 3.2. It is a matter of guessing which chord is in each section by dividing into time frames within the audio signal. The final chord recognition results are represented into sequence consisting of start time, end time, and chord symbol, as seen in Figure 3.3.

| **Start time** | **End time** | **Chord** |
|---|---|---|
| 0.00 | 10.00 | N |
| 10.00 | 11.94 | G# |
| 11.94 | 16.67 | D# |
| 16.67 | 17.41 | N |
| 17.41 | 19.17 | F# |
| 19.17 | 20.65 | B |
| 20.65 | 22.22 | F# |
| | ... | |

Figure 3.3: An example of chord recognition result.

### 3.1.2 Singing Melody Transcription

Automatic music transcription, which refers to converting an audio signal into the form of a symbolic score, is one of the most important research topics in the field of music information retrieval. Among the symbolic score forms, expressing notes with onset and offset time and pitch is referred to as note-level representation. Many attempts have been made to transcribe a singing melody into note-level representation, but this remains a difficult task. In vocal melodies, the onset and offset are often not apparent. Contrary to the exact pitch of instruments such as a piano, pitch vibrato

appears in various patterns depending on the style of the vocalist. Moreover, for a vocal melody in polyphonic audio with accompaniment, it is necessary to distinguish vocal timbres from mixed pitch patterns.

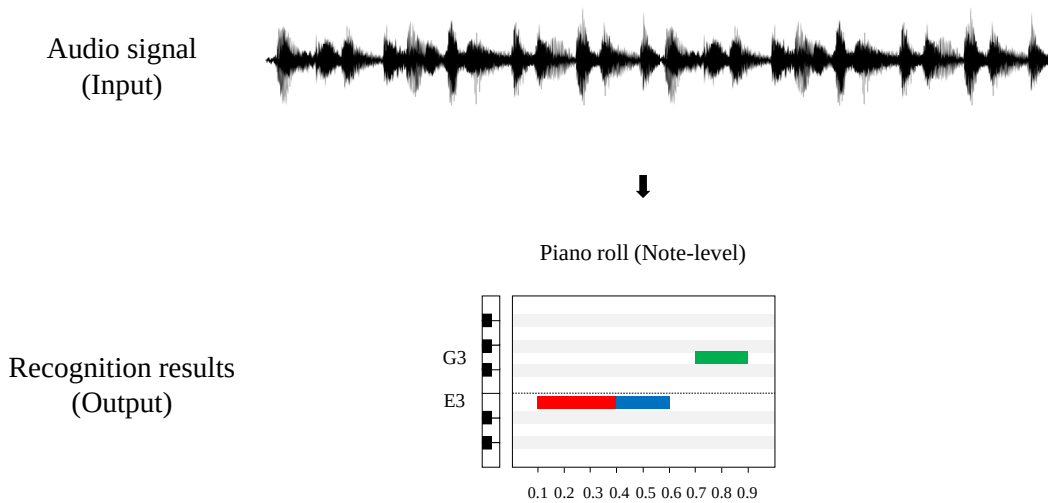



Figure 3.4: Problem definition of singing melody transcription.

The purpose of singing melody transcription is the recognize the monophonic vocal melody at note-level from popular music audio signals. It can be expressed visually, as in Figure 3.4. A single note consists of start tame, end time, and pitch, while pitch is recognized as MIDI pitch at the semitone level. The final singing melody transcription result is as seen in Figure 3.5.

### 3.1.3 Post-processing for Lead Sheet Representation

Since the results of the chord recognition and melody transcription are the time unit, in order to represent these as notes in lead sheet, additional information is required. Further, in order to convert each chord and melody into the notes used in sheet music, beat tracking is required. When using the beat tracking results, the

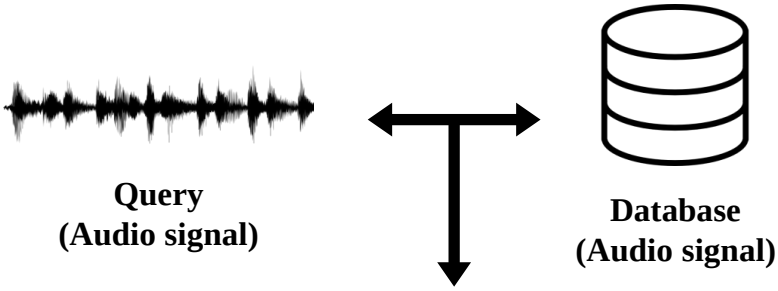
| Start time | End time | MIDI pitch |
|---|---|---|
| 6.29 | 6.49 | 58 |
| 6.49 | 7.19 | 60 |
| 8.95 | 9.27 | 58 |
| 9.27 | 10.23 | 58 |
| 11.83 | 12.19 | 56 |
| 12.19 | 12.35 | 60 |
| 12.35 | 12.97 | 60 |
| | ... | |

Figure 3.5: An example of singing melody transcription result.

time for each beat can be identified; thereby, chord and melody of time unit can be expressed into note of sheet music. Additionally, lead sheet needs to express musical key information, which requires key estimation from audio signals. Post-processing is a process of expressing the combined results of chord, melody, key, and beat recognition for representation in the lead sheet. Through this, music audio signals are converted into the final lead sheet form.

## 3.2 Melody Similarity Assessment

The purpose of the melody similarity assessment is to find the most similar section by evaluating the similarity of a popular music audio signal with the melody of other audio signals. This can be expressed visually, as shown in Figure 3.6. For melody similarity assessment, it is important to discover similarities of specific sections as opposed to the entire song. Therefore, it is a matter of finding sections that can be considered similar with respect to each section of the input song by comparing the melody of all the songs within the set to be compared. This problem setting can be used for melody plagiarism detection and cover song detection.




| Query section (sec) | Searched song | Section (sec) | Distance |
|---|---|---|---|
| 10 ~ 20 (s) | Artist A – Song a | 25 ~ 36 | 0.04 |
| 120 ~129 (s) | Artist B - Song b | 161 ~ 169 | 0.07 |
| 78 ~ 85 (s) | Artist C - Song c | 15 ~ 23 | 0.10 |
| | ... | | |

Figure 3.6: Problem definition of melody similarity assessment.

# Chapter 4

# A Bi-directional Transformer for Musical Chord Recognition

## 4.1 Methodology

### 4.1.1 Model Architecture

Making use of appropriate surrounding frames is essential for successful chord recognition [30, 74]. This context-dependent characteristic of the task is the motivation for applying the self-attention mechanism. With some modification to the original Transformer architecture, we present a bi-directional Transformer for chord recognition (BTC).[1]

[1] https://github.com/jayg996/BTC-ISMIR19

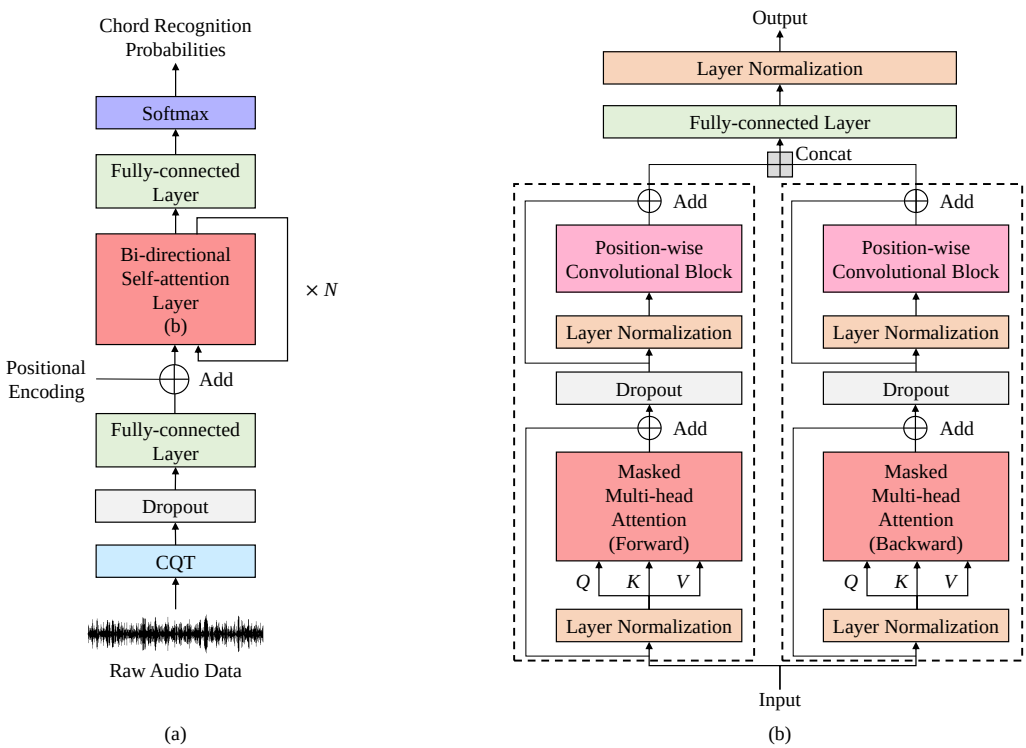



Figure 4.1: Structure of BTC. (a) shows the overall network architecture and (b) describes the bi-directional self-attention layer in detail. Dotted boxes indicate self-attention blocks.

The structure of BTC is shown in Figure 4.1. The model consists of bi-directional multi-head self-attentions, position-wise convolutional blocks, a positional encoding, layer normalization [75], dropout [76] and fully-connected layers. The model takes a CQT feature of 10 second audio signal (Section 4.2.2) as input. The results of adding positional encoding are given as input to two self-attention blocks with different masking directions, indicated as dotted boxes in Figure 4.1(b). The outputs are concatenated and are fed into a fully-connected layer so that the output size is the same as the original input. A stack of $N$ bi-directional self-attention layers is followed by another fully-connected layer that outputs logit values. The size of the logit values is the same as the number of chord labels. These logits are used to predict the chord and calculate the loss.

The loss function is a negative log-likelihood and all the model parameters are trained to minimize the loss given by the following equation (4.1).

$$L = -\sum_{t=1}^{T} \sum_{c \in V} y_c(t) log(\hat{y_c}(t)) \tag{4.1}$$

$T$ is the number of total time frames and $V$ is the chord label set. $y_c(t)$ is 1 if the reference label at time $t$ is $c$ and 0 otherwise. $\hat{y_c}(t)$ is the output of the model, representing the probability of the chord at time $t$ being $c$.

### Bi-directional Multi-head Self-attention

BTC employs multi-head self-attention as in the original Transformer. For each time frame, the input features are split into $n_h$ pieces and provided as input to the multi-head self-attention with the number of heads, $n_h$. Given $I$ as an input matrix, the multi-head self-attention can be computed as (4.2):

$$Multihead = Concat(head_1, ..., head_{n_h})W_O \tag{4.2}$$

$Q_j = (IW_Q)_j, K_j = (IW_K)_j$ and $V_j = (IW_V)_j$ are given as input to the attention function (2.1) to produce $head_j$ for $j = 1, ..., n_h$. $W_Q, W_K$ and $W_V$ are fully-connected layers that project the input to the dimension of $Q, K$ and $V$, respectively. $W_O$ is also a fully-connected layer that projects the concatenated output of dimension $(n_h \times d_{V_j})$ to the dimension of the final output. Dropout is applied to the softmax output weights when computing each $head_j$.

In BTC, self-attention can be interpreted as determining how much attention to apply to the value of the key time frame when inferring the chord of the query time frame. To prevent the loss of information due to the attention being performed to the entire input at once, we employed bi-directional masking. The forward / backward direction refers to masking all the preceding / succeeding time frames. The same masked multi-head attention module as the Transformer decoder was adopted. The bi-directional structure enables BTC to fully utilize the context before and after the target time frame.

Since the multi-head attention is performed on every time frame in the sequence, information about the order of the sequence is lost. We employed the same solution proposed by Transformer to address this issue: adding positional encoding results to the input, which are obtained by applying sinusoidal functions to each position. Since relative positions between two frames can be expressed as a linear function of the encodings, positional encoding helps the model learn to apply attention via relative positions.

#### Position-wise Convolutional Block

To utilize the adjacent feature information in a time frame, we replaced the position-wise fully-connected feed-forward network from the original Transformer architecture with a position-wise convolutional block. The position-wise convolutional block consists of a 1D convolution layer, a ReLU (Rectified Linear Unit) activation function and a dropout layer, where the whole sequence of layers is repeated $n_C$ times. Input and output channel size were identical to keep the feature size and sequence length constant. With the position-wise convolutional block, we anticipate to search the boundary and smooth the chord sequence by exploring adjacent information at each time frame.

### 4.1.2 Self-attention in Chord Recognition

For chord recognition, it is important to utilize not only the information from the target time frame but also from other related frames, which we call the context. The network architectures such as convolutional neural networks (CNNs) [77] or recurrent neural networks (RNNs) [78] can also explore the context, but self-attention is more suitable for the task because of the following reasons.

First, self-attention has selective usage of attention. In other words, the receptive field can be adaptive unlike CNNs where the kernel size is fixed. For example, assume that the labels for 16 frames are Cs for the first four frames, Gs and Fs for the next eight frames and Cs for the last four frames (see Figure 4.2). Consider the situation of recognizing Gs in frames 5 to 8. As for a CNN with kernel size of 3, when recognizing the chord of frame 7, the receptive field (frame 6 to 8) would be informative enough since all the frames contain the same chord. However, when inferring frame 5, the

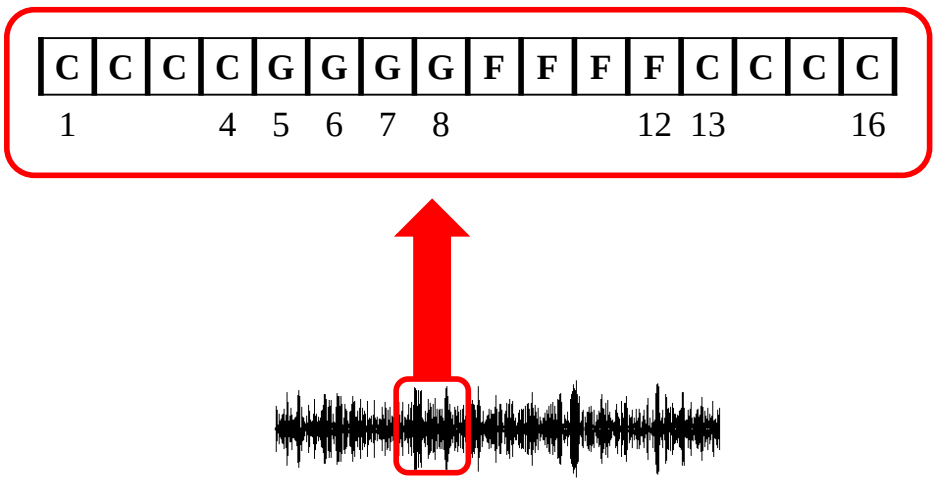



Figure 4.2: Chord sequence example

receptive field of frame 4 to 6 contains not only G but also C. With self-attention, on the other hand, the model can pay attention to the section of frame 5 to 8 regardless of the target frame's position.

Another advantage of attention mechanism is its ability to capture long-term dependency effectively. RNNs can also utilize distant information but direct access is not possible. For CNNs, there are two ways to access distant frames: by stacking layers in depth or by increasing the kernel size. The former has the same drawback as RNNs and the latter has the disadvantage that the weight sharing becomes less effective. Unlike these, self-attention has direct access to other frames no matter how far they are. Specifically, when recognizing the chord of frame 13, performing attention to first four frames would be helpful since they all contain C. With RNNs or deep CNNs, information that the first four frames were C would inevitably be diluted while passing through frames 5 to 12.

## 4.2 Experiments

### 4.2.1 Datasets

BTC and other baseline models were evaluated on the following datasets. A subset of 221 songs from Isophonics[2]: 171 songs by the Beatles, 12 songs by Carole King, 20 songs by Queen and 18 songs by Zweieck; Robbie Williams [79]: 65 songs by Robbie Williams; and a subset of 185 songs from UsPop2002[3]. These datasets consist of label files that specify the start time, end time and type of the chord. Due to copyright issue, these datasets do not include audio files. The audio files used in this work were collected from online music service providers (e.g. Melon[4]), which do not always provide the same audio files corresponding to the songs in the datasets. Since it was not possible to get exactly the same audio files, there were subtle differences in the chord start time of the label file and audio file. Accordingly we manually matched the labels to the audio file by shifting the whole label file back and forth, which resulted in no more than adding or deleting some "No chord" labels.

### 4.2.2 Preprocessing

Each 10-second-long audio signal (consecutive signals overlapping 5 seconds) was processed at the sampling rate of 22,050Hz using CQT with 6 octaves starting from C1, 24 bins per octave, and the hop size of 2048 [11]. The CQT features were transformed to log amplitude with $S_{log} = ln(S + \epsilon)$ where $S$ represents the CQT feature and $\epsilon$ is an extremely small number. After that, global z-normalization was applied with mean, variance from the training data.

[2] `http://isophonics.net/datasets`
[3] `https://github.com/tmc323/Chord-Annotations`
[4] `http://www.melon.com`

Pitch augmentation was also employed to the audio file with pyrubberband[5] package and labels were changed with pitch variation. Pitch augmentation between -5 $\sim$ +6 semitones were applied to all the training data.

Two different label types were used: maj-min and large vocabulary. The maj-min label type consists of 25 chords (12 semitones $\times$ {maj, min} and "No chord") [80]. The large vocabulary label type consists of 170 chords (12 semitones $\times$ {maj, min, dim, aug, min6, maj6, min7, minmaj7, maj7, 7, dim7, hdim7, sus2, sus4} and "X chord : the unknown chord", "No chord") [81]. From the label files, we extracted the chord that matches the time frame of input feature and transformed it to the appropriate label type.

### 4.2.3 Evaluation Metrics

The evaluation metric was weighted chord symbol recall (WCSR) score and 5-fold cross validation was applied to the entire data. When separating the evaluation data from the training data, there was no song included in both. The WCSR score can be computed as (4.3), where $t_c$ is the duration of correctly classified chord segments and $t_a$ is the duration of the entire chord segments.

$$WCSR = \frac{t_c}{t_a} \times 100(\%) \tag{4.3}$$

Scores were computed with mir_eval [82]. Root and Maj-min scores were used for the maj-min label type. Root, Thirds, Triads, Sevenths, Tetrads, Maj-min and MIREX scores were used for the large vocabulary label type. To calculate the score with mir_eval, the chord recognition results were converted into label files.

[5] `https://github.com/bmcfee/pyrubberband`

### 4.2.4 Training

| | | |
|---|---|---|
| Bi-directional self-attention layer | layer repetition ($N$) | {1, 2, 4, **8**, 12} |
| | self-attention heads ($n_h$) | {1, 2, **4**} |
| | dimension of $Q$, $K$, $V$ and all the hidden layers | {64, **128**, 256} |
| Position-wise convolutional block | block repetition ($n_C$) | **2** |
| | kernel size | **3** |
| | stride | **1** |
| | padding size | **1** |
| Dropout | dropout probability | {**0.2**, 0.3, 0.5} |

Table 4.1: Hyperparameters of BTC. Hyperparameters with the best validation performance are shown in bold.

Specific hyperparameters of BTC are summarized in Table 4.1. The hyperparameters with the best validation performance were obtained empirically after applying in 5-fold cross validation. Adam optimizer [83] was used with initial learning rate of $10^{-4}$. Learning rate was decayed with rate 0.95 when validation accuracy did not increase. Training was stopped if the validation accuracy did not improve for over 10 epochs.

## 4.3 Results

### 4.3.1 Quantitative Evaluation

Since existing studies of chord recognition were evaluated on different datasets, it is difficult to say that a particular model is the state-of-the-art. Among the models that were trainable with our datasets, we chose three baseline models with good performance: CNN, CNN+CRF and CRNN. CNN is a VGG [84]-style CNN and CNN+CRF has an additional CRF decoder [80]. CRNN is a combination of CNN and gated recurrent unit [85], named "CR2" in [81]. The input was preprocessed as mentioned in Section 4.2.2 for BTC and CRNN. For CNN+CRF and CNN, a single label was estimated with a patch of 15 time frames, in a similar way to [80].

Table 4.2 shows the performance comparison results of the baseline models and BTC for two label types. The best value for each metric is represented in bold. Among the models without a CRF decoder, BTC showed the best performance for all metrics. Including models with a CRF decoder, CNN+CRF obtained the best result in most of the metrics. Still, BTC shows comparable performance to CNN+CRF, performing better in Sevenths and Maj-min metrics for the large vocabulary label type.

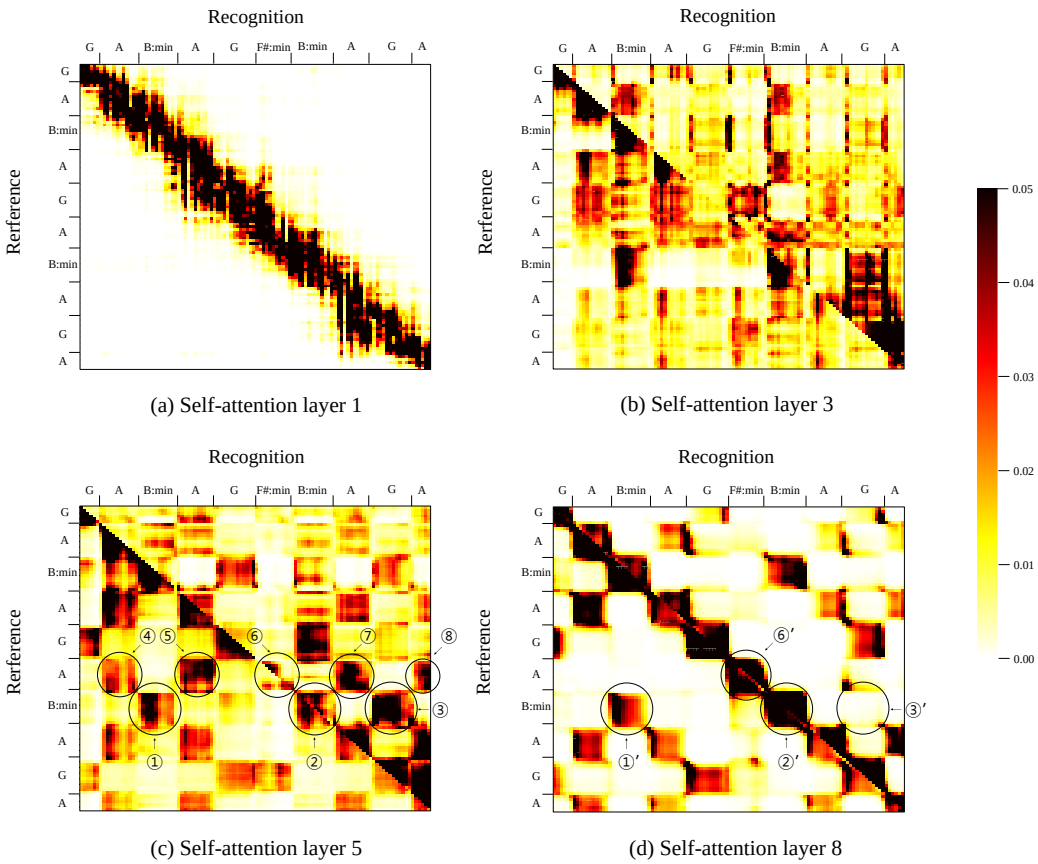
The main purpose of training a CRF decoder is to smooth the predicted chord sequences that are often fragmented. The performances of CRNN+CRF and BTC+CRF are also presented in Table 4.2 for comparison. Performance improvements due to the introduction of CRFs are evident in CNN but not in BTC and CRNN. This indicates that outputs of CNN were fragmented and an additional decoder training is necessary for better performance. On the other hand, BTC and CRNN can be trained with only CQT features and chord labels. That is, BTC requires only a single

training phase while achieving the performance comparable to that of CNN+CRF.

| Model | maj-min label type | | large vocabulary label type | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Root | Maj-min | Root | Thirds | Triads | Sevenths | Tetrads | Maj-min | MIREX |
| CNN | 83.6$_{\pm 1.3}$ | 81.8$_{\pm 1.2}$ | 83.5$_{\pm 1.4}$ | 80.4$_{\pm 1.2}$ | 75.5$_{\pm 0.6}$ | 71.5$_{\pm 1.9}$ | 65.2$_{\pm 1.0}$ | 81.9$_{\pm 1.4}$ | 79.8$_{\pm 0.7}$ |
| CNN+CRF [80] | **84.0**$_{\pm 1.3}$ | **83.1**$_{\pm 1.4}$ | **83.7**$_{\pm 1.5}$ | **81.1**$_{\pm 1.4}$ | **76.3**$_{\pm 0.8}$ | 71.3$_{\pm 1.9}$ | **65.7**$_{\pm 1.6}$ | 82.1$_{\pm 1.5}$ | **81.8**$_{\pm 1.1}$ |
| CRNN [81] | 83.4$_{\pm 0.8}$ | 82.3$_{\pm 0.9}$ | 82.9$_{\pm 1.1}$ | 80.1$_{\pm 1.0}$ | 75.3$_{\pm 0.7}$ | 71.3$_{\pm 1.9}$ | 65.2$_{\pm 0.9}$ | 81.5$_{\pm 1.3}$ | 79.9$_{\pm 0.8}$ |
| CRNN+CRF | 83.3$_{\pm 0.8}$ | 82.3$_{\pm 1.0}$ | 82.7$_{\pm 1.2}$ | 79.7$_{\pm 0.9}$ | 74.8$_{\pm 0.5}$ | 69.5$_{\pm 2.0}$ | 63.9$_{\pm 1.0}$ | 80.7$_{\pm 1.4}$ | 80.2$_{\pm 1.0}$ |
| BTC | 83.8$_{\pm 1.0}$ | 82.7$_{\pm 1.0}$ | 83.5$_{\pm 1.2}$ | 80.8$_{\pm 1.0}$ | 75.9$_{\pm 0.5}$ | **71.8**$_{\pm 1.7}$ | 65.5$_{\pm 0.9}$ | **82.3**$_{\pm 1.2}$ | 80.8$_{\pm 0.9}$ |
| BTC+CRF | 83.9$_{\pm 1.0}$ | **83.1**$_{\pm 1.1}$ | 83.5$_{\pm 1.2}$ | 80.7$_{\pm 1.1}$ | 75.7$_{\pm 0.5}$ | 70.7$_{\pm 2.0}$ | 64.8$_{\pm 1.1}$ | 81.7$_{\pm 1.4}$ | 81.4$_{\pm 0.9}$ |

Table 4.2: WCSR scores averaged over the same 5 folds. Numbers next to the scores denote the standard deviations.

### 4.3.2 Attention Map Analysis




(a) Self-attention layer 1

(b) Self-attention layer 3

(c) Self-attention layer 5

(d) Self-attention layer 8

Figure 4.3: The figures represent the probability values of the attention of self-attention layers 1, 3, 5 and 8 respectively. The layers that best represent the different characteristics were chosen. The input audio is the song "Just A Girl" (0m30s ∼ 0m40s) by No Doubt from UsPop2002, which was in evaluation data.

Attention maps demonstrate that each self-attention layer has different characteristics. Figure 4.3 shows the attention map of self-attention layers 1, 3, 5 and 8, trained with the maj-min label type. The lower / upper triangle of each attention map represents the attention probability of the forward / backward direction self-attention layer. The labels of the vertical axis and the horizontal axis are the reference chord and the chord recognition result of the target time frame, respectively. The cell of $i$-th row and $j$-th column represents the attention probability to the $j$-th time frame when inferring the chord of the $i$-th time frame.

At the first self-attention layer, only neighboring frames are used to construct the representation of the target frame. For the third layer, the attention is widely spread over all time frames, yet still with higher probabilities for nearby frames than distant frames. At the fifth layer, several adjacent time frames form a group, which appears in a rectangular region in the attention map. This means that the model divides the whole input into some sections, which is possible due to the adaptive receptive field. The network focuses only on a few important sections to identify the target frame, regardless of the distance between section and the frame. Unlike the fifth layer, attention is more dense in certain regions at the eighth layer. In particular, the boundary of the high probability region matches that of the final recognition result.

Specifically, at the fifth layer in Figure 4.3(c), the reference chord for region ② is B:min. Region ① shares the same reference chord B:min and the network assigns high attention probabilities to region ① for time frames in region ②. This phenomenon is similar in layer 8 between ①′ and ②′(Figure 4.3(d)), which results in the correct final chord recognition of B:min. In contrast, for region ③ where the

reference chord is G, the attention probability is high at layer 5 but not for region ③′ at layer 8. This can be attributed to G and B:min sharing two notes in common, since G and B:min consist of (G,B,D) and (B,D,F#) respectively. In other words, attention at layer 5 can be seen as attention to partial features of chords sharing the same notes. None the less, the final recognition result after the last layer is not G but B:min. This is possible because of the multi-head attention structure: the other heads might lower the attention probability even if the attention to a wrong chord is active, leading to the correct result.

On the other hand, there are cases where the recognition results are wrong in a similar situation. The reference chord for regions ⑥ and ⑥′ is A. At layer 5, the attention mechanism seems to work well with high attention probabilities to region ④,⑤,⑦ and ⑧, where the reference chords are all As. However, the attention to those regions cannot be seen at the last layer, and the final recognition result is not A but F#:min. This recognition failure can be regarded as a result of two notes of F#:min (F#,A,C#) overlapping with A (A,C#,E).

To summarize, for each target frame in the input audio, the model uses only neighboring frames at first. At the middle layers, the model gradually broadens the receptive field and selectively focuses on time frames with characteristics similar to that of the target frame. Finally, at the last layer, the attention is performed on only essential information for chord recognition.

# Chapter 5

# Note-level Singing Melody Transcription

## 5.1 Methodology

We propose a sequence-to-sequence note-level singing melody transcription Transformer and some techniques to improve the performance.[1] In Section 5.1.1, a novel note event token set is defined to express a monophonic melody as a token sequence. Section 5.1.2 describes audio features which are given as input to the model. Section 5.1.3 introduces the overall model structure with its difference from the original Transformer, while Section 5.1.4 describes the inference and masking strategy to enforce the output sequence to be monophonic. Section 5.1.5 to Section 5.1.7 introduce three techniques for effective singing melody transcription respectively, namely overlapping decoding, pitch augmentation, and adding noisy dataset with data cleansing.

### 5.1.1 Monophonic Note Event Sequence

In this paper, a monophonic melody is represented as a sequence of musical event tokens. Each token in the event token set belongs to one of the following types: time, pitch, start of sequence (SOS), EOS, or padding (PAD).

By fixing the length and time resolution of an audio segment at $N$ seconds and 10 ms, respectively, the number of time frames in the segment is $T = 100 \times N + 1$,

[1] `https://github.com/jayg996/IDA-Singing-Melody-Transcription`

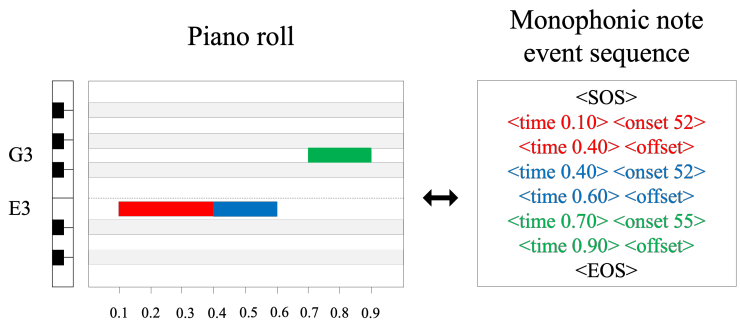



Figure 5.1: An example of a monophonic note event sequence. Time tokens indicate the absolute positions of the events in the audio segment. A pitch token represents either an onset event of one of the 128 MIDI pitch numbers or an offset event.

which equals to the number of different absolute time tokens. As in [27], 128 pitch onset tokens are used, each symbolizing the onset of a MIDI pitch from 0 to 127. The difference is that a single offset token represents the offset of all 128 MIDI pitches, thereby enforcing the melody to be monophonic. If the onset of a note occurs immediately after the offset of the previous note, the offset event precedes the onset event. Finally, SOS and EOS tokens are added to the beginning and the end of the sequence, respectively. PAD token is used after EOS token to equalize the length of the sequences in a mini-batch. An example of the monophonic note event sequence is depicted in Figure 5.1.

### 5.1.2 Audio Features

The magnitude values of STFT were utilized as the input audio representation. The audio sample rate was 16 kHz, and the window size and the hop length of STFT were 2,048 and 160, respectively. The length of the unit time frame was 10 ms. The STFT parameters previously used in singing melody F0 estimation [1] were referenced.

In training phase, the audio signal is randomly cropped into sections of $N$ sec-

onds. A single time frame can correspond to various time tokens through random cropping, so it is possible to provide various training data.

### 5.1.3 Model Architecture

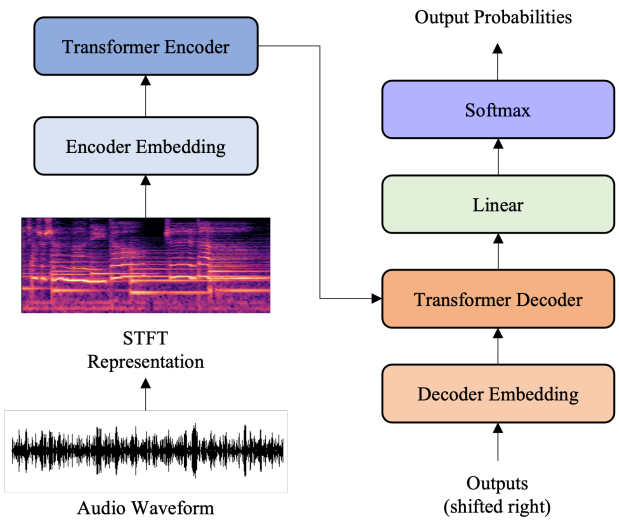



Figure 5.2: Overall structure of the proposed note-level melody transcription model. Transformer encoder and decoder are similar to those of the original Transformer [2].

The structure of the proposed note-level singing melody transcription model is depicted in Figure 5.2. While [27] adopted T5 [86] as the network architecture, our model resembles the original Transformer [2] with some modifications to the embedding layers.

Since the input of the encoder is STFT, layer normalization [87] is first applied to normalize STFT magnitude values. After adding sinusoidal positional encoding, another layer normalization is applied given that the warm-up stage can be omitted by pre-normalization [88]. Figure 5.3 shows the resulting embedding layer architectures of encoder and decoder.

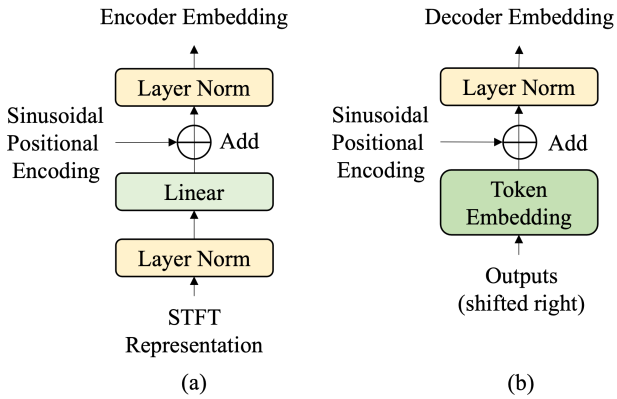



Figure 5.3: Detailed structures of (a) encoder and (b) decoder embedding layers. In the encoder, layer normalization is first applied to normalize the STFT magnitude values. Both encoder and decoder embedding layers apply layer normalization after adding the positional encoding.

### 5.1.4 Autoregressive Decoding and Monophonic Masking

At the inference phase, the event tokens are decoded autoregressively. The encoder receives an audio signal of $N$ seconds as input, and the decoder autoregressively predicts the next token, starting from SOS until EOS.

Several maskings are applied when computing the subsequent token probabilities, to ensure that the recognition result is a monophonic melody. Time and pitch tokens are forced to be decoded alternately by masking one type after another. When time tokens are to be predicted, tokens that indicate the previous time are masked. For the prediction sequence to end within a limited length, the last token of the decoder output is forced to be EOS.

### 5.1.5 Overlapping Decoding

In [27], non-overlapping audio segments were recognized separately and the results were combined to transcribe longer audio signals. Such non-overlapping decoding

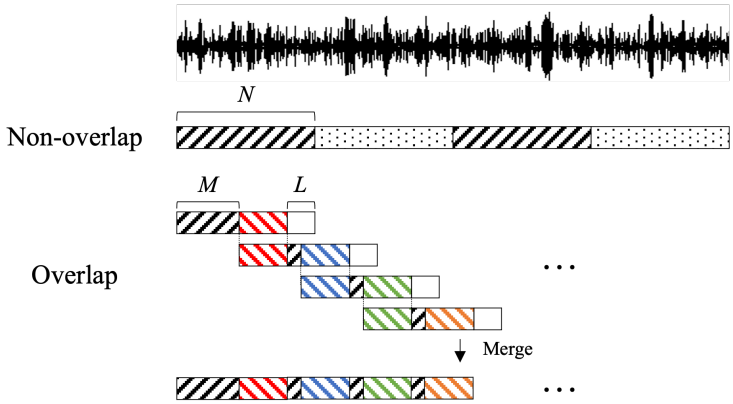



Figure 5.4: Illustration of the difference between overlapping and non-overlapping decoding. $N$, $M$, and $L$ are the input length of the model, the hop size, and the length to be omitted, respectively.

has a problem in that the context of the previous segment is lost. For example, if a segment is truncated after a note onset but before its offset, decoding should be performed in the next segment without the note's onset information.

We propose overlapping decoding in this paper to overcome the limitation. With overlapping decoding, successive segments overlap for a certain length of time. This prevents the context from being disconnected by transferring some of the results recognized in the previous segment to the next segment. Among the notes recognized in the previous segment, notes that overlap with the next segment are replicated to the next segment by modifying the time tokens to match the absolute time within the next segment. These notes act as a prior sequence when autoregressively inferring the next segment's notes. And to avoid discontinuity in the transcription results, a certain length of time in the end of the overlapping region is discarded and inferred again in the next segment.

As depicted in Figure 5.4, the hop size between segments is $M$ seconds, where

$M \leq N/2$. The events recognized in the last $L$ seconds are discarded where $L < M$, and the window of length $N$ moves on to the next overlapping segment. The first $M$ seconds of the prediction sequence is stored for the entire sequence, and the succeeding $N-(M+L)$ seconds is used as the prime sequence for the next segment. Accordingly, it is possible to transcribe the audio signal using the context of the previous segment.

### 5.1.6 Pitch Augmentation

As training a deep learning model requires a large amount of data, data augmentation is one of the most common attempts to improve performance [89]. Various augmentations such as pitch shifting and time stretching has been widely adopted in previous studies in the field of MIR [90]. Especially in AMT, since acquiring the pair of music audio and high-quality label data is very costly and time consuming, data augmentation is one attractive option to enlarge training data.

To be more specific, pitch augmentation refers to shifting the entire pitch of an audio clip several semitones up or down. Through pitch augmentation, various pitch tokens can be uniformly exposed during the training process. This prevents the output probability distribution of pitch tokens from being biased to some common tokens, resulting in less overfitting and generalization of the model.

In this work, a Python library designed to apply effects to the audio signal, *pysndfx* [2], is used to augment pitch of audio. Pitch augmentation is randomly applied only during training, from -6 to +6 semitones, to both the audio and label data.

[2] `https://github.com/carlthome/python-audio-effects`

### 5.1.7 Adding Noisy Dataset with Data Cleansing

A large amount of training data is required to develop a model with robust performance, but obtaining high-quality labeled data is laborious. Although DALI [91] is a noisy dataset with incorrect labels, some of the songs are labeled correctly, and it would be a more valuable dataset if one could distinguish between the correct and incorrect songs. When examining the DALI dataset, it turned out that the most common label errors were octave error and time shift. Therefore, we manipulated the label in terms of octave and time shift and compared it with the F0 estimation [1] result, and classified it as data that can be used for training if it exceeds a threshold.

Specifically, data cleansing was performed by shifting the annotation in both pitch and time axes and comparing with the recognition result of F0 estimation [1]. The sliding window sizes were 1 octave and 10 ms for pitch and time, respectively, in the ranges of -2 $\sim$ +2 octaves and -5 $\sim$ +5 seconds. For a song, if the maximum raw pitch accuracy of F0 estimation among the shifted candidates was lower than 0.6, the song was discarded.

## 5.2 Experiments

### 5.2.1 Dataset

In this paper, two public datasets were used: MIR-ST500 [12] and DALI [91]. MIR-ST500 is a dataset with note-level annotations of vocal melodies for polyphonic audio signals. It consists of 500 songs, and only 474 songs were available at the time of the experiments. The dataset was split into three sets: songs numbered from 1 to 350, 351 to 400, and 401 to 500, which were used for training, validation, and testing of the experiments, respectively. To enable direct comparison with previous studies [12, 48], we used the same data split as the publication of the dataset. As for the test data split, all 100 songs were available without missing data, and was used for the ablation study and comparison with other models.

DALI is another note-level singing melody annotation dataset for polyphonic audio signals. It is the largest public singing transcription dataset currently available. A total of 4,927 songs were available, but the dataset has many incorrect labels because it was annotated automatically [12, 92]. Therefore, data cleansing described in Section 5.1.7 was applied. Consequently, 858 songs were left, which were used only for training to verify the effect of the additional noisy dataset.

MedleyDB [93], an F0 dataset which differs from a note-level dataset, was also evaluated for performance evaluation and comparison. The test data split of [94] was adopted, and only 12 songs were used as in [1, 49]. The F0 annotations of vocal melody in polyphonic audio were used as labels.

### 5.2.2 Experiment Configurations

Specific hyperparameters of Transformer are summarized in Table 5.1 and other configurations used in the experiment are as follows. The model was trained with cross entropy loss function. The Adam optimizer was adopted, with an initial learning rate of 0.0001 and a batch size of 12. The learning rate was decayed with a factor of 0.5 if the validation loss did not decrease for more than 3 epochs, and the experiment was terminated if the loss did not decrease for 10 epochs. The number of time tokens was 1,025, enabling representation of 0 to 10.24 seconds with a time resolution of 10 ms. Adding 128 pitch onset tokens and offset, SOS, EOS, and PAD tokens to the token set results in a total of 1,157 tokens. The duration of the audio input $N$ was fixed to 5.12 seconds during training and inference. In overlapping decoding, the hop size $M$ and the length of the last part to be discarded $L$ were 2.56 and 1.28 seconds, respectively.

Table 5.1: Hyperparameters of Transformer.

| | | |
|---|---|---|
| Transformer encoder | number of layers | 8 |
| | embedding dimension | 512 |
| | self-attention heads | 8 |
| | dimension of query, key, value | 512 |
| | hidden size of feed-forward networks | 1024 |
| | dropout probability | 0.1 |
| Transformer decoder | number of layers | 8 |
| | embedding dimension | 512 |
| | self-attention heads | 8 |
| | dimension of query, key, value | 512 |
| | hidden size of feed-forward networks | 1024 |
| | dropout probability | 0.1 |
| | maximum length | 512 |

### 5.2.3 Evaluation Metrics

The transcription metrics of *mir_eval* [82] were used for the evaluation of note-level singing melody transcription. Four types of metrics were selected: onset time, offset time, onset with pitch, and note-level which considers all of the onset, offset, and pitch. A threshold was set according to each criterion to evaluate the transcription results, and is considered correct if the difference between the predicted value and the groundtruth is less than the threshold. In this paper, the thresholds for onset time, offset time, and pitch were 50 ms, $max(50 \text{ ms}, 0.2 * \text{note duration})$, and 50 cents (= 0.5 semitone), respectively. For each metric with different criteria, the recall (R), precision (P), and F1 score (F) were all evaluated.

Additionally, F0 estimation evaluation metrics were used to evaluate the voice detection and pitch-only transcription performance. The melody metrics of *mir_eval* were used as the evaluation metrics for F0 estimation. Note-level labels and predictions were converted into F0 sequences with the time resolution of 0.01 seconds. For time frames not included in any note, the frequency was set to 0 Hz (unvoiced). Voicing recall rate (VR) and voicing false alarm rate (VFA) were used as metrics to evaluate voice detection. Raw pitch accuracy (RPA) and raw chroma accuracy (RCA) were used as metrics to evaluate pitch tracking. Overall accuracy (OA) was used as a metric to evaluate the performance of voice detection and pitch tracking simultaneously. The threshold to judge the correctness of the pitch was set at 50 cents. Equations (5.1)-(5.5) are defined to compute each metric. The number of voiced frames and the total number of frames in the reference are denoted by $v$ and $t$, respectively. $\hat{v}_c$, $\hat{p}_c$, and $\hat{c}_c$ are the number of correctly predicted frames for voice detection, pitch, and chroma, respectively. $\hat{v}_{ic}$ is the number of frames incorrectly

predicted as voiced frames and $\hat{t}_c$ is the number of all frames correctly predicted.

$$VR = \frac{\hat{v}_c}{v} \tag{5.1}$$

$$VFA = \frac{\hat{v}_{ic}}{t - v} \tag{5.2}$$

$$RPA = \frac{\hat{p}_c}{v} \tag{5.3}$$

$$RCA = \frac{\hat{c}_c}{v} \tag{5.4}$$

$$OA = \frac{\hat{t}_c}{t} \tag{5.5}$$

### 5.2.4 Comparison Models

EfficientNet-b0 [12] was chosen as a comparative model to train and test on MIR-ST500 dataset. The metrics were computed from the public prediction results of the test dataset released by the authors. JDC$_{note}$ [48] is a model trained with the labeled MIR-ST500 and additional unlabeled datasets through self-training. The experiment results reported for the test set of MIR-ST500 were compared directly.

Tony [40] and Omnizart [43], which are public note-level singing transcription models, were also selected as comparative models. The transcription result of Tony, a public software, was analyzed by exporting the result to MIDI. Vocal audio separated using Spleeter [6] was given as an input to Tony because the performance of singing transcription dropped significantly for polyphonic audio. The singing transcription result of Omnizart was obtained using a public source code library.

For the comparison model of vocal melody F0 estimation, JDC [1] was chosen. It can recognize a vocal melody from polyphonic audio with its voice detection module. The pre-trained model shared by the authors was used for performance evaluation.

### 5.2.5 Human Evaluation

In order to analyze whether our proposed model achieved significant performance improvement, we asked people to evaluate the results. The transcription result was converted into a MIDI piano sound source and was played along with the original audio. In addition, the piano roll was provided as an image so that the results of transcription could be visually evaluated.

A total of three transcription results were evaluated: ground truth, EfficientNet-b0 [12], and the propsed model. Since ground truth is the most accurate transcription result, it was used as a criterion for accurate transcription when people listened to it and evaluated it. EfficientNet-b0 was selected as a comparison model because it showed the highest note-level F1 score among comparison models. In the test dataset of MIR-ST500, 140-160 seconds of 10 songs (410.mp3, 420.mp3, ..., 490.mp3, 500.mp3) were used. For the same section of 10 songs, the results of three models were provided in random order so that people could listen and evaluate the performance.

The criteria for evaluating performance were evaluated in terms of note onset, offset, pitch, and overall. The transcription performance was scored on a 5-point scale ranging from 1 (poor) to 5 (good) for each criterion. Experimental subjects were recruited from Amazon Mechanical Turk [95], and only the results of those who evaluated the ground truth as the highest overall average score were collected for the reliability of the experiment. As a result, the results evaluated by 32 people were collected.

## 5.3 Results

### 5.3.1 Ablation Study

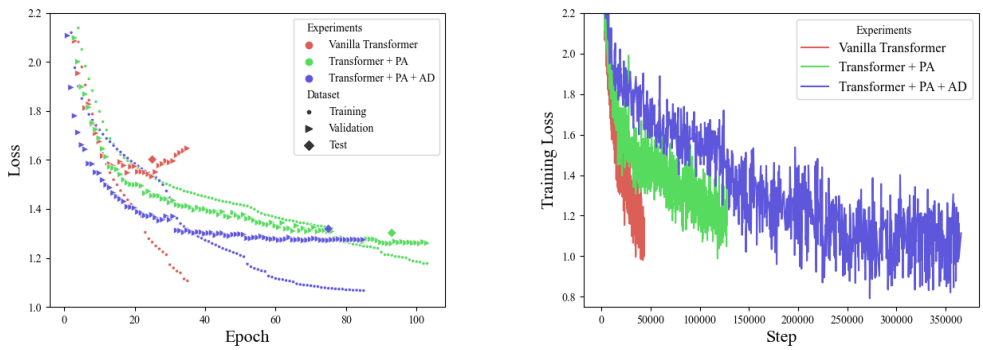
The experimental results of note-level singing transcription are summarized in Table 5.2. Ablation studies were conducted to confirm the effects of overlapping decoding (OD), pitch augmentation (PA), and adding noisy dataset with data cleansing (AD).

First, the F1 score of note-level is improved by 0.013 by introducing OD. The performance improvement in the offset F1 score is more noticeable than onset, which is plausible because the onset of the previous segment is no longer lost. For non-overlapping decoding, determining an offset event is problematic because it is not possible to know whether the pitch onset event has occurred in the previous segment.

Adding PA led to a significant improvement in note-level F1-score by 0.09. PA increases the amount of training data due to exposure to pitch classes that do not appear frequently, preventing overfitting. Figure 5.5 (a) implies the relevance between PA and overfitting. The validation loss increases after 25 epochs for the vanilla Transformer, implying overfitting. In contrast, although the training loss decreased slowly with PA, the validation loss continued to decrease without overfitting.

One of the notable results is that by including DALI dataset in the training data, the performance of the note-level F1 score improved by 0.01. Even though training and testing on only DALI resulted in poor performance, AD demonstrated a performance improvement. The effect of DALI dataset can also be found in Figure 5.5. Figure 5.5 (b) demonstrates the training loss decreasing more slowly with AD. Moreover, as illustrated in Figure 5.5 (a), the validation loss continuously decreased along with the training loss. AD is beneficial because it prevents the model from memorizing the training data and generalizes the model performance.

Although the label data of MIR-ST500 are accurate, adding PA and AD were effective in performance improvement. Since PA and AD have the effects of increasing the training data, we expect that the performance can be further improved with larger datasets.




(a) (b)

Figure 5.5: (a) Change in loss according to the epoch for each data split in each experiment. (b) Training loss according to the number of steps. Red indicates Transformer (baseline), green indicates + pitch augmentation, and blue indicates +noisy training dataset (proposed).

Table 5.2: Performance evaluation results of models for MIR-ST500 dataset

| Model | Onset | | | Offset | | | Onset + Pitch | | | Note-level | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F | P | R | F |
| Vanilla Transformer | 0.752 | 0.714 | 0.730 | 0.692 | 0.655 | 0.670 | 0.700 | 0.665 | 0.679 | 0.492 | 0.468 | 0.478 |
| Transformer + OD | 0.737 | 0.729 | 0.731 | 0.688 | 0.679 | 0.681 | 0.690 | 0.684 | 0.685 | 0.494 | 0.490 | 0.491 |
| Transformer + OD + PA | 0.771 | 0.779 | 0.774 | 0.745 | 0.752 | 0.747 | 0.739 | 0.748 | 0.742 | 0.578 | 0.585 | 0.581 |
| Transformer + OD + PA + AD (Proposed) | **0.785** | **0.791** | **0.787** | **0.747** | **0.753** | **0.749** | **0.755** | **0.761** | **0.757** | **0.589** | **0.595** | **0.591** |
| EfficientNet-b0 [12] | 0.742 | 0.778 | 0.754 | 0.625 | 0.655 | 0.637 | 0.654 | 0.686 | 0.666 | 0.448 | 0.472 | 0.458 |
| $JDC_{note}$ [48] | - | - | 0.762 | - | - | - | - | - | 0.697 | - | - | 0.422 |
| Tony [40] | 0.542 | 0.595 | 0.564 | 0.562 | 0.619 | 0.587 | 0.415 | 0.453 | 0.431 | 0.262 | 0.288 | 0.273 |
| Omnizart [43] | 0.428 | 0.489 | 0.454 | 0.436 | 0.504 | 0.464 | 0.329 | 0.374 | 0.348 | 0.173 | 0.197 | 0.183 |

### 5.3.2 Note-level Transcription Model Comparison

The results of comparing different state-of-the-art note-level singing transcription models with MIR-ST500 test dataset are also reported in Table 5.2. The proposed model outperformed the comparison models for all the metrics considered. There was a significant performance improvement in which the note-level F1 score increased by 0.1 or more compared with other models. This can be attributed to the performance improvement in the sequence-to-sequence model structure and the methods specialized for note-level singing transcription. Tony and Omnizart achieved poor performance because they were not trained with MIR-ST500. Compared with EfficientNet-b0, vanilla Transformer achieved a higher offset F1 score. Consequently, predicting the musical note sequence in the decoder is more advantageous than predicting for every time frame because the offset of the vocal melody is often ambiguous.

### 5.3.3 Transcription Performance Distribution Analysis

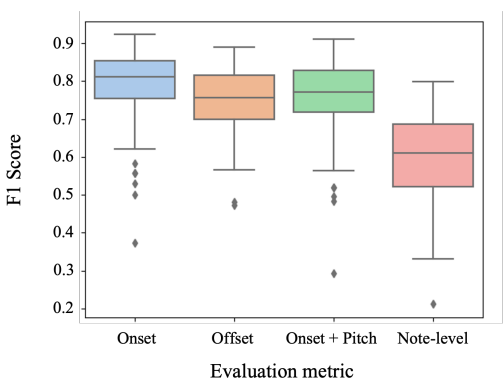



Figure 5.6: Box plots representing the distribution of the proposed model performance on MIR-ST500 test dataset. The y-axis indicates the F1 score.

The proposed model's evaluation results on MIR-ST500 test data are visualized in Figure 5.6 as box plots. The onset prediction achieved a higher F1 score than the

offset prediction, supporting the assumption that offset in vocal melody is ambiguous. Compared with predicting only the onset, adding pitch prediction resulted in a lower F1 score, which is predictable. The difference between the two is insignificant, implying that the pitch prediction can be considered accurate if the onset prediction is successful. In contrast, the note-level F1 score was noticeably low because note-level prediction requires accurate prediction of onset, offset, and pitch for a single note. Moreover, the transcription performance varies significantly depending on the song. Regarding note-level prediction, the proposed model successfully transcribed one song with the highest F1 score of 0.8 while reporting the worst performance of 0.21 for another. Some of the plausible reasons why the results vary widely depending on the song are discussed in detail in Section 5.4.2.

### 5.3.4 Fundamental Frequency (F0) Metric Evaluation

Table 5.3: F0 evaluation results of the proposed model and JDC [1]

| Dataset | Model | VR | VFA | RPA | RCA | OA |
|---|---|---|---|---|---|---|
| MIR-ST500 | Proposed | **0.907** | 0.144 | **0.848** | **0.849** | **0.851** |
| | JDC | 0.780 | **0.110** | 0.586 | 0.590 | 0.708 |
| MedleyDB | Proposed | **0.800** | 0.128 | 0.493 | 0.493 | 0.696 |
| | JDC | 0.774 | **0.117** | **0.719** | **0.726** | **0.818** |

Table 5.3 presents the results evaluated by F0 metrics. For metrics related to voice detection, the proposed model performed better in VR, and JDC performed better in VFA. This result can be interpreted as caused by training the proposed model to achieve a high recall rate, increasing false alarm rate. However, the difference in VFA between the proposed model and JDC is subtle, implying that the voice detection of the proposed model is comparable to JDC.

For pitch-related metrics, namely RPA and RCA, and the overall performance

metric OA, the results were contradictory depending on the dataset. With MIR-ST500, the proposed model outperformed JDC, whereas with MedleyDB, JDC achieved superior results likely caused by the difference between the annotation and prediction method. For example, the results of note-level and F0 annotations may exhibit significant differences in vibrato notes or note transitions with dragging pitch. A vibrato note is covered over several semitones with F0 annotation, but in note-level, it is annotated as a single pitch level. For note transitions with dragging pitch, F0 annotation expresses each pitch change in detail, whereas only two notes are remained at note-level. In such cases, the evaluation results are likely superior when the annotation and prediction coincide. Furthermore, whereas RPA and RCA differ by more than 0.4% for JDC, the proposed model exhibits almost no difference, suggesting that the proposed model commits fewer octave errors.

## 5.4 Qualitative Analysis

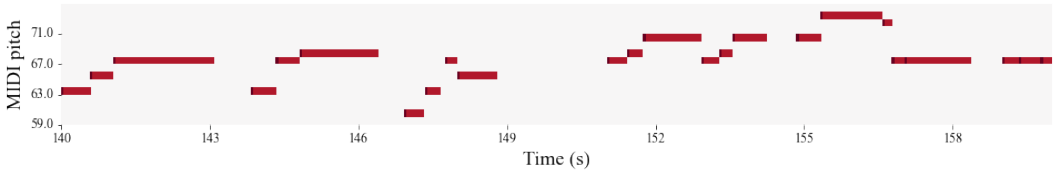
### 5.4.1 Visualization of Ablation Study

Through qualitative ablation studies, we analyzed the effects of introducing OD, PA, and AD to vanilla Transformer. Figure 5.7 is a visualization of the transcription results for a test song in the MIR-ST500 dataset. While Figure 5.7 (a) represents the ground truth label of singing melody transcription, (b), (c), (d), and (e) are images expressing the ground truth label and the recognition results together.

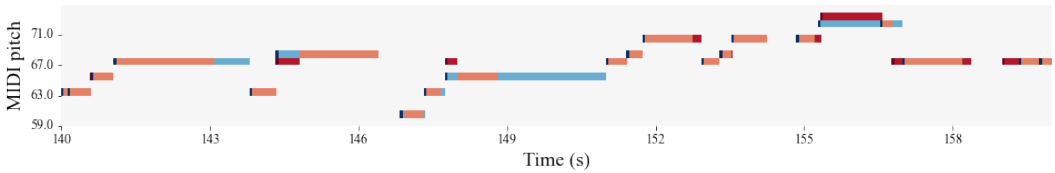
Figure 5.7 (b) and (c) are the transcription results of the same model but different decoding strategy. The former is the result using non-overlapping decoding, while the latter is the result of applying OD. The biggest difference between the two is the offset of notes. In Figure 5.7 (b), regarding the two notes at 143 and 149 seconds, respectively, the notes do not end and continue until the onset of the next note. On the other hand, in Figure 5.7 (c), the offset of the corresponding notes were predicted after few time frames, resulting in a relatively accurate transcription. The reason for missing the note offset in non-overlapping decoding is that the presence of a note onset in the previous segment is unknown due to the context loss problem. Through the proposed OD, the context loss problem was mitigated and note offsets were captured.

Figure 5.7 (d) is the transcription result of the model with PA added. Compared to Figure 5.7 (c), the timing of the onset in the notes around 140 seconds is slightly more accurate. And for the note at 156 seconds, pitch transcription with PA was correct whereas the model without PA predicted the wrong pitch. These visible differences explain the significant improvement of the evaluation metrics in Table 5.2.

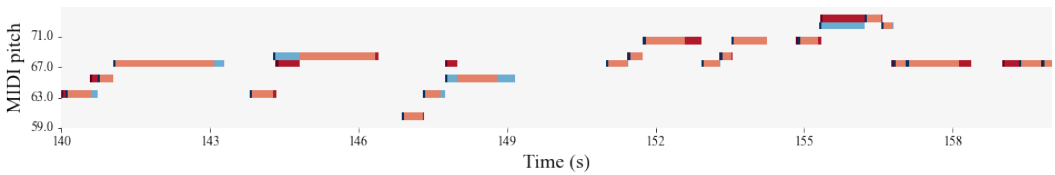
Figure 5.7 (e) is our proposed model applying OD, PA, and AD altogether. The notes at 144, 148, and 153 seconds, which were all incorrect notes in Figure 5.7 (d), were recognized correctly. Most of the recognition results match the ground truth label, and there are no well-marked blue colored notes except the note at 149 seconds. The example shows that the proposed methods are effective in improving transcription performance, in accordance with the quantitative results in Section 5.3.1.



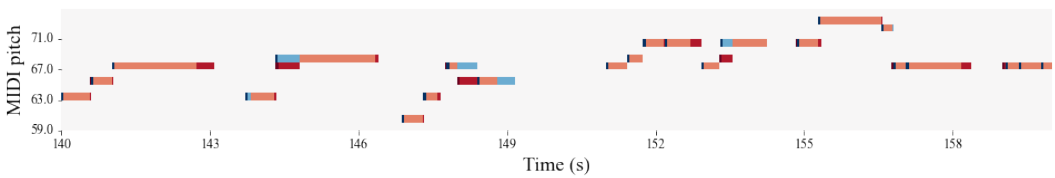

(a) Ground truth



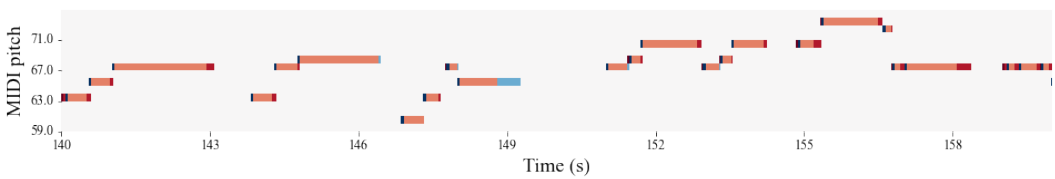

(b) Vanilla Transformer




(c) Transformer + OD

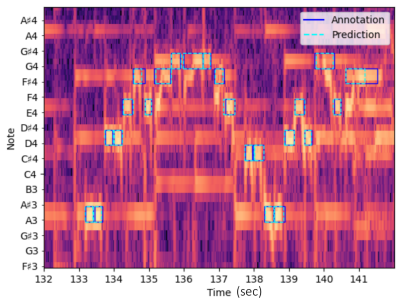



(d) Transformer + OD + PA

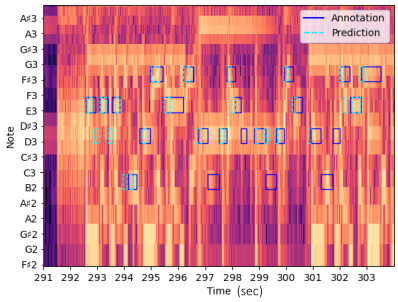



(e) Transformer + OD + PA + AD (Proposed)

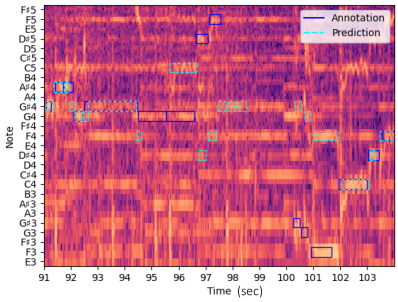
Figure 5.7: Visualization of the transcription results of "460.mp3" in the MIR-ST500 test dataset. The onset is indicated in dark color for each note. In (b), (c), (d), and (e), the ground truth label, prediction, and the shared part are shown in red, blue, and orange, respectively.




(a) Song id 405




(b) Song id 470




(c) Song id 404

Figure 5.8: Visualizations of note-level singing melody transcription results for test examples from MIR-ST500 dataset. The STFT representation is expressed as a spectrogram. The annotated labels and prediction results are depicted as solid blue lines and dotted cyan lines, respectively, according to the pitch and time of the notes. In (a), most of the prediction results and correct annotations are consistent, and in (b) and (c), they are not.

### 5.4.2 Spectrogram Analysis

In analyzing the results in more detail, some examples of spectrograms of test songs along with the annotated labels and prediction results are visualized in Figure 5.8. Figure 5.8 (a) illustrates the spectrogram of the best transcription results with an F1 score of 0.8. Most notes were accurately predicted with respect to onset, offset, and pitch, except the offset of the last note. The result is explainable through several aspects of the music audio: the vocal voice is audibly clear, the accompaniment

sound is relatively calm, and there are few chorus voices.

In contrast, Figures 5.8 (b) and (c) are examples of poor performance. In Figure 5.8 (b), most of the notes' onsets and offsets are inaccurate, with some notes even missing. The low F1 score of 0.33 can be explained as caused by the vocal's whisper-like singing style, obscuring the onsets and offsets of the singing notes.

Some specific pitch prediction errors are examined in Figure 5.8 (c), in which the F1 score was 0.35. One of the most common prediction errors was the octave error. For example, for the notes at 97 and 101 seconds, the model predicted the pitch as D#4 and F4, whereas the ground truth pitch labels were D#5 and F3. Moreover, the prediction result of the G4 note at 96 seconds was C5: the pitch class itself was incorrect. One reasonable explanation is that because the chorus vocal is heavily inserted in the song, the loud chorus was the predominant cause of pitch inaccuracy. At 98 seconds, there were some non-existent notes in the prediction results, likely because the model detected instrument sound as the vocal melody.

Based on analyzing the examples of the prediction results, the results are more accurate when the singer's voice is clear and the chorus and instrument sounds are quiet. In contrast, because the proposed model is a monophonic singing melody transcription model, the performance was poor for multi-vocal audio.

### 5.4.3 Human Evaluation

Table 5.4: The average scores evaluated by humans for the results of each model. Numbers next to the scores denote the standard deviations.

| Model | Onset | Offset | Pitch | Overall |
|---|---|---|---|---|
| Ground truth | 3.90 $\pm 0.53$ | 3.95 $\pm 0.67$ | 4.08 $\pm 0.52$ | 4.19 $\pm 0.47$ |
| Proposed | 3.89 $\pm 0.56$ | 3.80 $\pm 0.67$ | 4.00 $\pm 0.59$ | 3.86 $\pm 0.53$ |
| EfficientNet-b0 [12] | 3.79 $\pm 0.68$ | 3.78 $\pm 0.67$ | 3.99 $\pm 0.62$ | 3.82 $\pm 0.51$ |

The results of human evaluation of singing melody transcription can be seen in Table 5.4. The score of ground truth, which is an accurate transcribed answer, is the highest in all aspects. Our proposed model showed the second best performance in all aspects, following ground truth. In particular, the proposed model regarding onset received a score of 3.89, close to the ground truth's 3.90, which is significantly ahead of EfficientNet-b0. For offset, pitch, and overall scores, the proposed model achieved slightly higher scores than EfficientNet-b0.

In terms of overall scores, a notable gap still remains between ground truth and AMT models. In order to be recognized as perfect transcription by humans, performance improvement through additional research and data collection is required.

# Chapter 6

# Automatic Music Lead Sheet Transcription

## 6.1 Post-processing for Lead Sheet Representation

Chapter 4, 5's research results are utilized for chord recognition and singing melody trancription. [54] is used for musical key estimation model. It is a model of recognizing key with respect to the music audio signals using deep convolutional network. The estimated key is classified into 24 types, major and minor, for 12 musical scales, and one key representing the entire song is recognized. [60] is applied for a beat tracking model, which uses recurrent neural network and deep bayesian network model. Time signature can be estimated to 3/4 and 4/4 and it recognizes at what seconds the beat within each bar exists. Downbeat by each bar is represented as 1 and it repeats as many as the number of beats within in order.

Chord and melody are time units and to convert this into the length of note represented in transcription, beat needs to be utilized. The thesis set the minimum unit of chord notation as a quarter note while the minimum unit of melody notation is sixteenth note. In the case of chords, chord symbol, which is most frequent time within the length of quarter note is determined as the chord for the beat. As for the melody, each quarter note is exactly divided into four to make semiquaver, moved to the closest beat time at the start and end of note, and was represented in note. As

for the melody, each quarter note is divided into four to make sixteenth notes, and the start and end times of each note are moved to the nearest beat time. This can be expressed visually, as shown in Figure 6.1. Through post-processing, the music audio signals are finally converted into lead sheet.

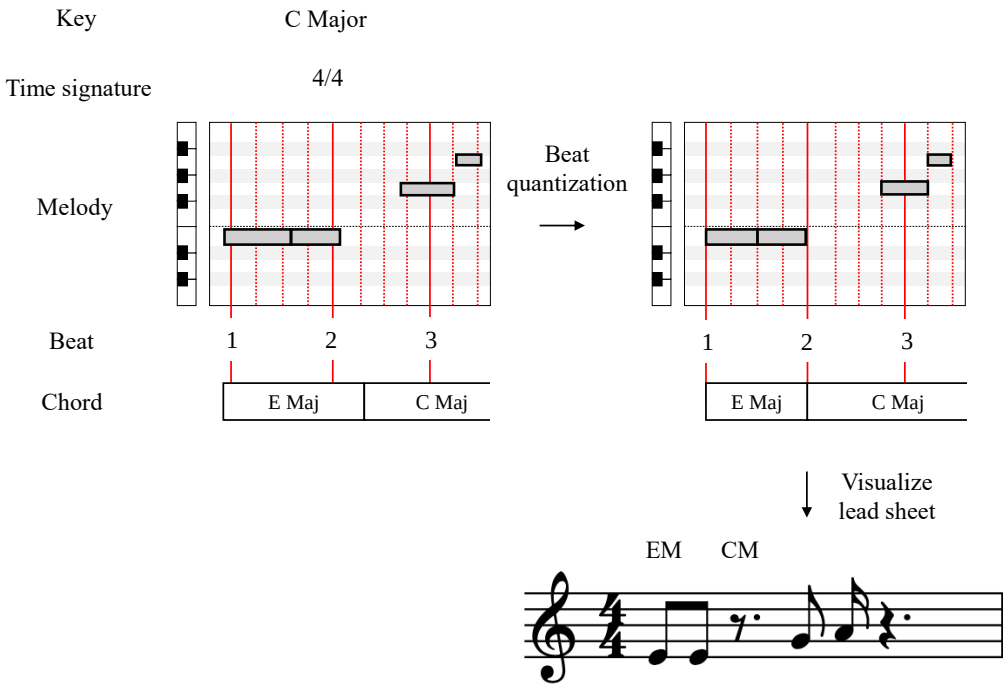


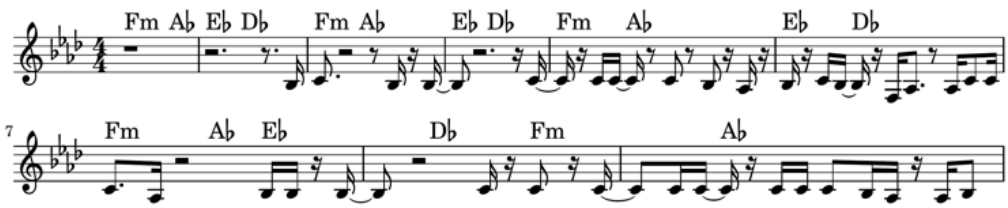

Figure 6.1: It is a process of transcribing using the key, chord, beat and melody information recognized from audio signals. The red line is four quarter notes while the red dotted line is semiquaver notes.

## 6.2 Lead Sheet Transcription Results

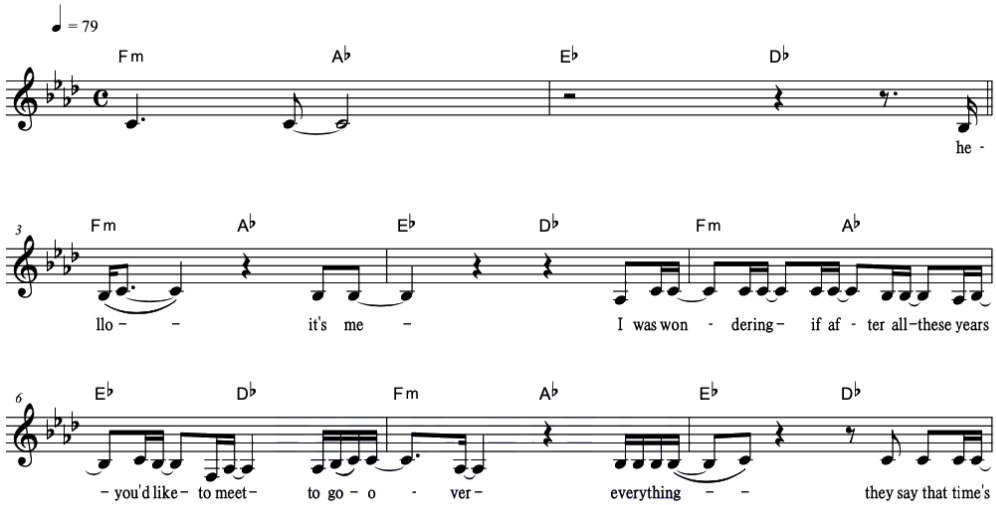
There is no metric to evaluate the results of converting the music audio signals into lead sheet. Since there are labels for the chord, key, beat, and melody recognition, it is possible to extract the numerical performance in the middle stage. However, it is vague to set a metric to evaluate after combining all the recognition results and converting it into lead sheet through post-processing. In addition, there is an issue with difficulty of identifying if a wrong part in the final lead sheet form is from the error of which recognition model. Therefore, we intend to use the actual case of lead sheets transcribed from audio signals and lead sheets transcribed by experts to compare. Thereby, the thesis aims to conduct a qualitative evaluation of the performance of automatic lead sheet transcription and identify which area needs complementation.

The first case of analysis is shown in Figure 6.2. (a) is the result of automatic lead sheet transcription while (b) is the lead sheet transcribed by an expert. Since the methods of visualizing transcriptions are different, the number of bars within a single stave is different. Although the thesis does not transcribe lyrics, generally as shown in (b), lead sheet usually contains lyric information. First, we can identify that for key signature, both are represented as the same (Ab Major). Time signature is also expressed with the same as 4/4. In the case of chord sequence, we can identify the repletion of Fm-Ab-Eb-Db as identical. However, from the 7th bar of (a), the beat of chord is different from the downbeat. This can be seen as the difference between the recognition of the timing of the beat and the timing of the chord change. As for the melody, (a) and (b) are very different. Generally, what appears to be the most different is the rhythm of the melody. In (a), it shows that there are a lot of rests

between consecutive melody notes, but compared to (b), it can be seen that this was misrecognized. They were recognized as rests since the distances among the notes were long enough among during the beat quantization process at post-processing. This is due to the accuracy of predicting the note's onset and offset times. As for the order of the melody's pitch, aside from the beat, most of them seems correct.



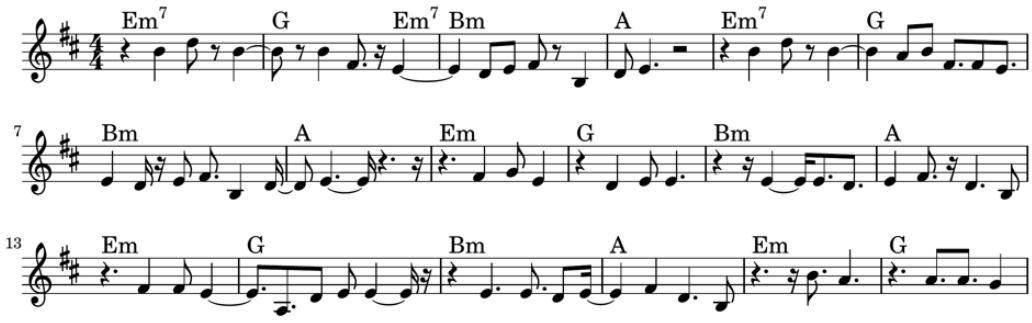

(a)



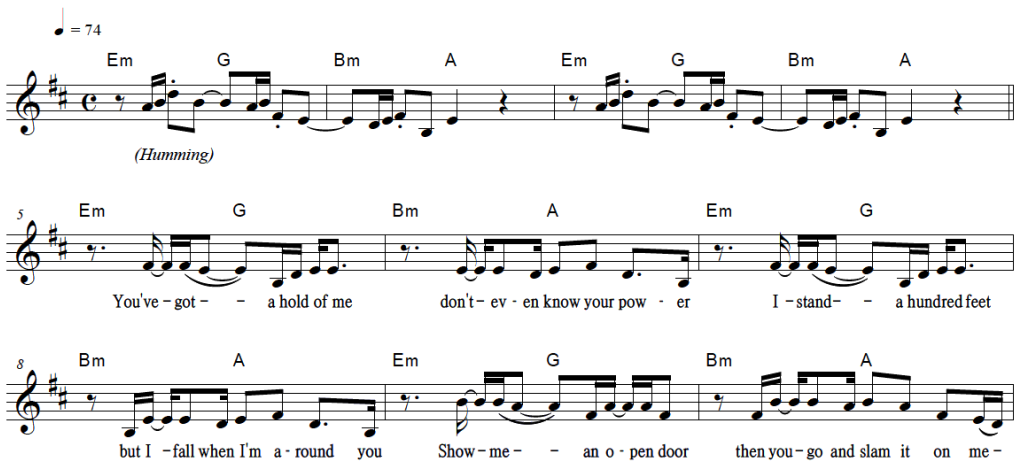

(b)

Figure 6.2: (a) is a transcription result of automatic music lead sheet transcription, while (b) is a transcription by a music expert.

The second case of analysis is shown in Figure 6.3. Key signature is expressed the same, as D major. Time signature is also represented the same as four-four time. However, looking at the timing of chord and measure change, (a) seems to recognize bpm (beats per minute) twice as fast as (b). Such an error is referred to as an octave error. It indicates that one actual beat was recognized as split in half. Because of this, while (b) shows two chords for each bar, (a) has one chord per bar in general. Some details seem different for the chord recognition results. (b) represented a chord as Em while (a) recognized it as Em7. In addition, while the second bar of (a) recognizes the two chords of G and Em7, the last two beats of the first bar in (b) are represented with a single G chord. It appears that the chord recognition model recognized the chords in more detail. As for the melody, due to the result of the beat tracking, (a) seems to have about twice the long notes as (b). In addition, the rhythms are overall different since notes are recognized as being divided and the length of the rests are largely different. As for the pitch of melody, the overall flow is consistent but the note's rhythm is condensed, ignoring the details of the pitch. As the recognition performance for the onset, offset, and pitch of each melody note improves, it can be accurately transcribed.




(a)

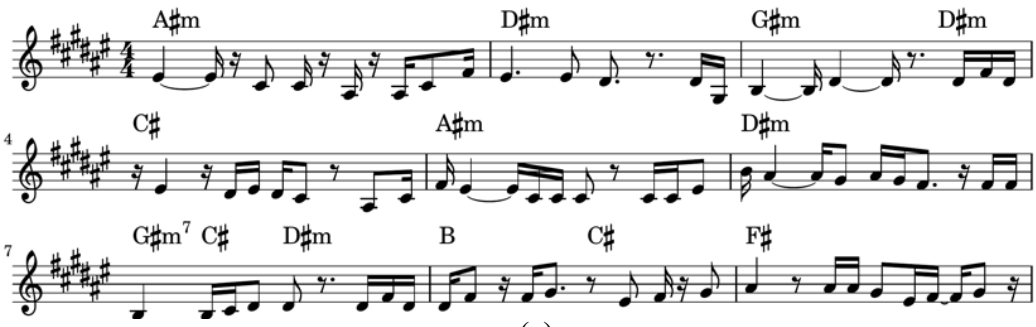



(b)

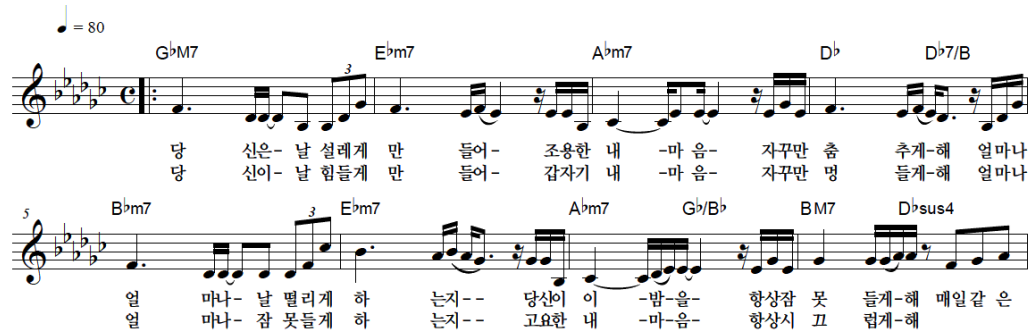
Figure 6.3: (a) is a transcription result of automatic music lead sheet transcription, while (b) is a transcription by a music expert.

The third case of analysis is shown in Figure 6.4. Since the key signature expressed in (a) and (b) are different, the overall transcriptions are represented entirely differently. For instance, while (a) recognizes the key signature as F# Major, (b) displays it as Gb Major. The two keys are entirely identical in the perspectives of pitch; therefore, it should be regarded as a different in representation. As for the chord recognition results, there is not only the difference in representation, but the actually recognized chords are also different. The first bar in (a) is recognized as A#m while (b) represented it as GbM7. As for the rest of the chords, most of them are displayed differently due to the difference between sharp and flat. As for the melody, similar to the previous two cases, overall flow of the pitch is similar between (a) and (b). However, as the notes' rhythms differ, there were differences such as notes that should have been divided being combined or unnecessary rests being included. In particular, when looking at the last three notes in the first bar in (b), it is expressed as triplet. However, the last three notes in the first bar of (a) is entirely different. This can be seen as a result of the failure to express triplet since during the post-processing process, the minimum beat unit was set as 16th note.

The thesis conducted an overall comparison between the lead sheet by an expert and lead sheet transcribed using our method. In most cases, key and chord did not have much difference. However, the part with the biggest difference from the actual transcriptions was the pitch and rhythm of the melody. In order to improve this, the performance of the singing transcription model recognizing melody's onset, offset, and pitch should be improved. Further, the accuracy of the beats used when converting it into transcriptions should also be improved. In addition, the post-processing also requires delicacy to represent special cases such as triplets.




(a)




(b)

Figure 6.4: (a) is a transcription result of automatic music lead sheet transcription, while (b) is a transcription by a music expert.

# Chapter 7

# Melody Similarity Assessment with Self-supervised Convolutional Neural Networks

## 7.1 Methodology

### 7.1.1 Input Data Representation

Using the results of Chapter 6, in order to address the problem of melody similarity assessment, chord and key are excluded and only melody is used. The minimum beat unit of the melody is sixteenth note, and pitch is MIDI pitch, expressed from 0 to 127. In order to express the melody as an image, it is converted into the form of piano roll with x-axis being the time axis of sixteenth note unit while y-axis is the pitch axis of semitone unit. The piano roll expresses the value as 1 if the pitch is played at a certain time and as 0 if it is not played. In the piano roll, when the note of the same pitch continues, the note's breakpoint cannot be identified. Therefore, the onset roll is also used as the data representation, with the onset time of starting point of the note expressed as 1 while the rest is expressed as 0. In order to use both piano roll and onset roll, it is expressed to overlap with different channels on the image.

Since melody similarity assessment needs to take place by section, the evaluation of the similarity on specific section should be possible. Therefore, instead of using the melody for the entire song at once, the sections are divided by a certain length.

Although the standard for similar melody is not certain in plagiarism case [16], we presume that if four bars are very similar, it can be considered as similar melody. In the pre-processing procedure, based on the starting point of each bar, four bars are cut and used as sections. As a result, the shape of each data input is 64 (time) x 128 (pitch) x 2 (channel). When representing this as a image, it is as shown in Figure 7.1.

Further, simple filtering methods are used to remove the meaninglessly similar cases. First, when the number of notes is less than four in four bars, the part is not used. In addition, if there is an empty bar among the four bars, the section is excluded. Through filtering, similar cases due to lack of melody can be excluded.

### 7.1.2 Data Augmentation

Since we do not have as labels whether the melodies are similar, it is necessary to create a positive sample with some transformation. In order to provide the data of similar melody, data augmentation is used. Each augmentation method can be expressed visually, as shown in Figure 7.2. First, the key shift is augmentation moving pitches of all notes within the four bars simultaneously from -7 to +7 semitones. Since the entire melody moves together, it can be seen as an identical melody.

As for the augmentation to be applied for each note, six augmentation methods are applied: add note, delete note, merge note, note pitch shift, change note duration, and note split. As for the add note, one of the pitches of the note is selected and a part among the rest section is selected to add note. Delete note is a method of randomly selecting a note and deleting it. The merge note is a method of modifying into playing as the pitch of the note selected, such as from the starting point of the note prior to the selected note to the ending point of the selected note. The note

pitch shift is a method of modifying the pitch of the selected note into the pitch of other notes within the four bars that is randomly selected. Regarding the change note duration, it is a method of randomly adjusting the duration of the selected note to the extent that it does not affect other notes. Note split is a method of randomly dividing the entire duration of the selected note into two notes and pitches are selected one among different notes' pitches.

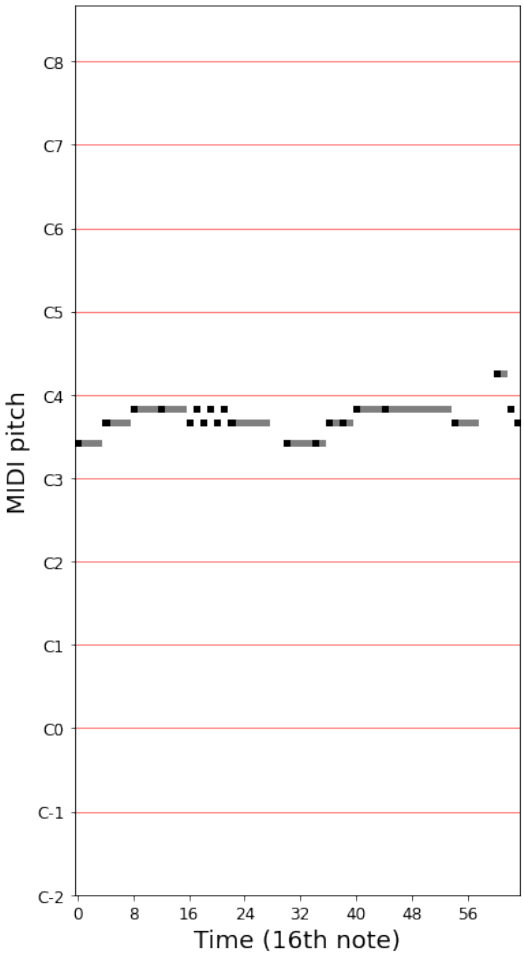



Figure 7.1: In order to distinguish the starting frame and continuing frame of notes, input representation was made with two channels, piano roll and onset roll. In the image, onset is displayed in black, while the frame which is not an onset but where the note continues is displayed in grey.

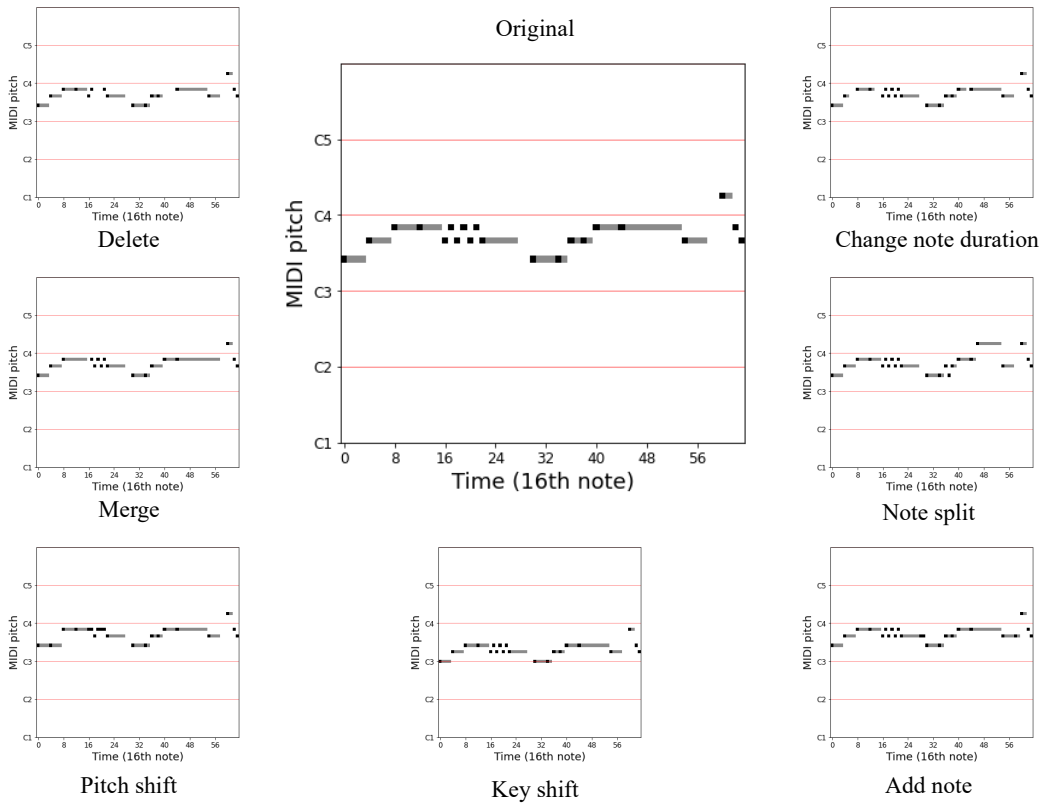



Figure 7.2: This is the result of applying each data augmentation on original input data. We can identify that there are parts similar to the given melody and that some are modified.

For training data, anchor, key-shifted anchor, positive sample, and negative sample are prepared. The anchor is a melody of four bars from a song. The key-shifted anchor is the result of changing only the key in the anchor. The positive sample is created through the augmentation method defined above. First, anchor or key-shifted anchor is randomly selected. Then, among the entire number of notes, notes to which an augmentation method is applied are selected at a ratio from the minimum $r$ to the maximum $R$. One of the methods among the six note augmentation methods is randomly selected to be applied on each of the notes for augmentation. The result of this is a positive sample. As for a negative sample, it is a melody of four bars that come from a song completely unrelated to the anchor.

### 7.1.3 Model Architecture

In order to map the input image on embedding space, ResNet [68] structure is used. The detailed structure of the model is seen as Figure 7.3. The model utilizes convolutional layer, batch normalization, ReLU activation, and residual connection repeatedly, reducing and summarizing the size of the entire image. Lastly, channel, width, and height axes are all flattened out to apply fully connected layer moving toward 256 dimension's embedding space. In addition, in order to restrict the size of the embedding space, normalization is used to make the size of the embedding vector 1.

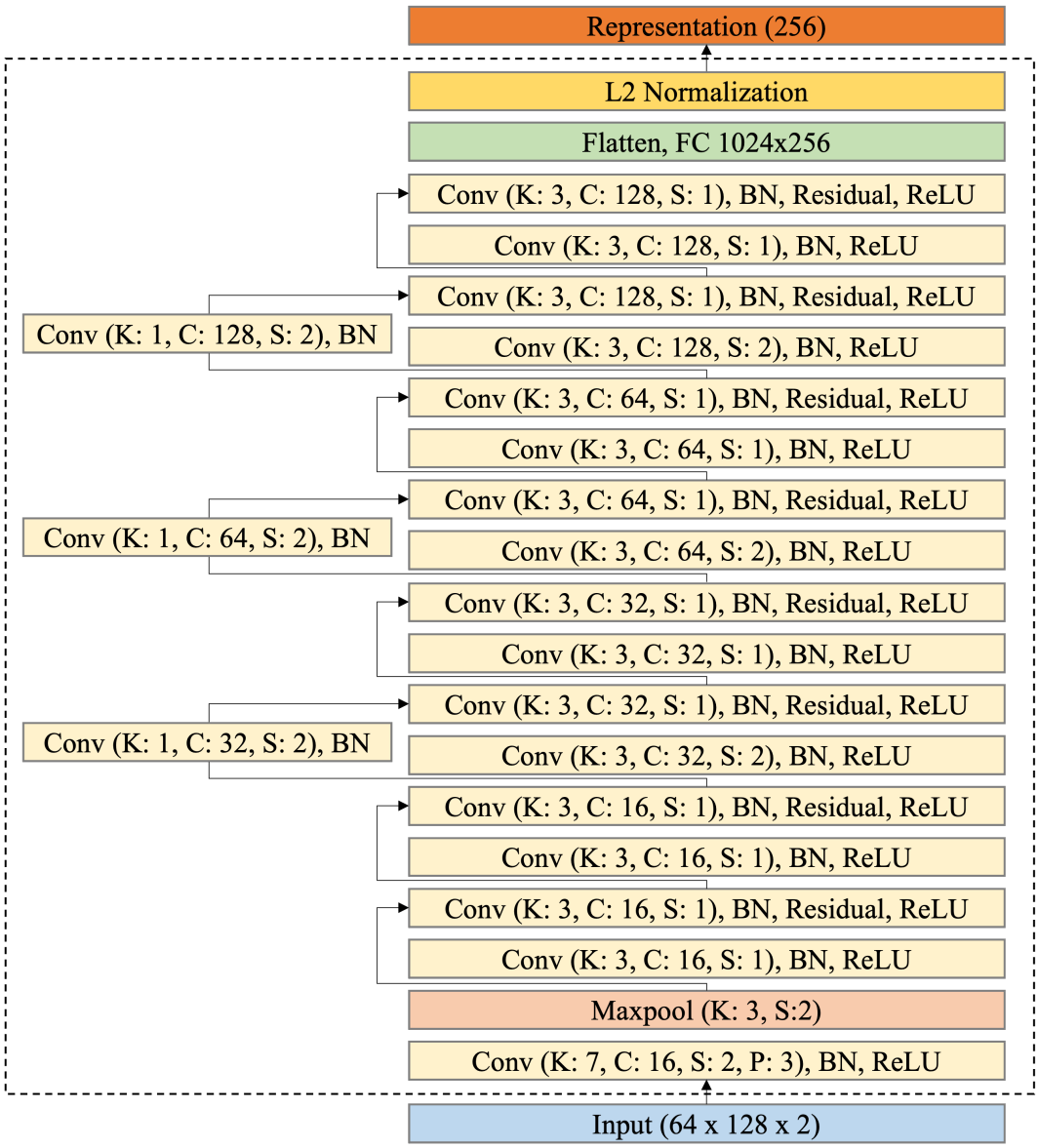



Figure 7.3: ResNet like Network structure. K: kernel size, C: channel number, S: stride, P: padding. Padding is omitted for layers with 1x1 padding. The output dimension is 256, which is used as an embedding vector for the input.

### 7.1.4 Loss Function

Similar to [96], since it is leaning only using musical augmentation on the given data without label information, it can be considered self-supervised learning. For model training, triplet loss [70] is used. At Section 7.1.2, anchor, key-shifted anchor, positive sample, and negative sample are prepared in the data preparation process.

Since just changing key is not considered to change the melody, anchor and key-shifted anchor need to be represented the closest. Further, as for the positive sample with changes in a few notes, it should be represented to be farther than the key-shifted anchor but closer than the negative sample. Since the negative sample is a melody irrelevant from anchor, it should be represented the furthest.

For this, triplet loss is separately used for each case. First, key-shifted anchor is set as positive sample while positive sample is set as negative sample, making positive sample further than key-shifted anchor. Here, margin is set at $\alpha$. Next, positive sample is used as positive sample, while negative sample is used as negative sample with margin set at $\beta$. Finally, key-shifted anchor is set as positive sample, while negative sample is set as negative sample with margin set at $\gamma$. In order to express the sequential relationship for the four types of data in the loss, the margin values are set to $\alpha + \beta = \gamma$. The entire loss is the sum of all three losses and, this can be expressed as Equation 7.1.

$$Loss = Loss_1 + Loss_2 + Loss_3 \tag{7.1}$$

$$Loss_1 = Triplet(a, k, p, \alpha) \tag{7.2}$$

$$Loss_2 = Triplet(a, p, n, \beta) \tag{7.3}$$

$$Loss_3 = Triplet(a, k, n, \gamma) \tag{7.4}$$

$a$ is anchor, $k$ is key-shifted anchor, $p$ is positive sample, $n$ is negative sample, and $\alpha$, $\beta$, $\gamma$ are margin values. The triplet loss function is Equation 2.2. This is represented visually, as shown in Figure 7.4.

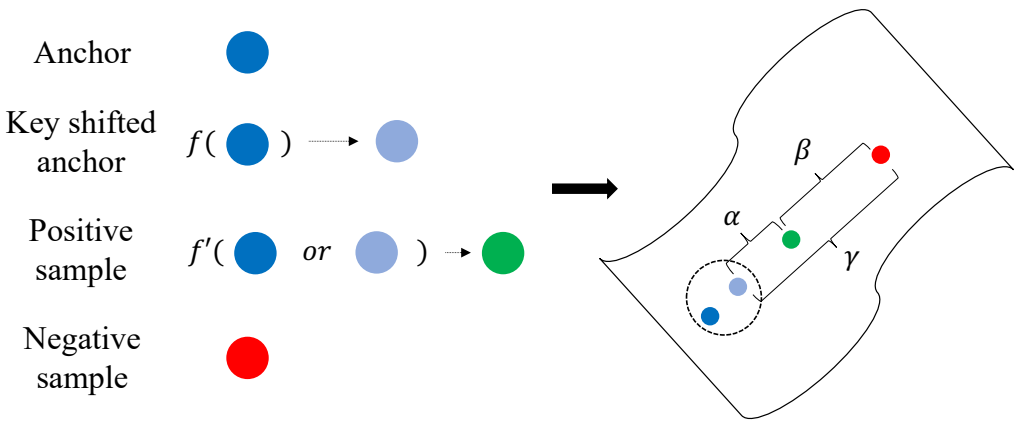



Figure 7.4: An overview of our self-supervised learning approach. We construct training data by transforming anchor with randomized augmentation functions.

### 7.1.5 Definition of Distance between Songs

For plagiarism detection and cover song detection, the distance-based searching method is as follows. First, with respect to the query song, data pre-processing method in Section 7.1.1 is applied identically to create all the image of melody of four bars unit. Next, with respect to the search object songs, the data pre-processing is applied the same to create an image of four bars unit. All images are mapped to embedding vectors using the model. With respect to all embeddings of a query song, compare all the embeddings of search objects and measure the distance. Define the distance between the pair with the shortest distance between the query song and search object song.

$$Distance(X, Y) = \min_{i,j} \| f_{model}(x_i) - f_{model}(y_j) \|_2^2 \tag{7.5}$$

$X$ and $Y$ are songs while $f_{model}$ is the deep learning model. The search result of query song is created by arranging search object songs in the order of closeness.

## 7.2 Experiments

### 7.2.1 Dataset

For model training, a custom dataset of 14,669 songs was used. Each song is a popular song with singing melody and other instruments. Applying the result of Chapter 6, each song was converted into the lead sheet form. The entire dataset was divided into the train: validation: test with the ratio of 0.8:0.1:0.1.

To evaluate the performance of trained model's performance, Plagiarism dataset and Cover song dataset were established. After collecting all of the audio files, applying the result of Chapter 6, each song was converted into the lead sheet form. First, Plagiarism dataset was made into a total 10 pairs of plagiarism songs. The dataset was established with cases where the melody was plagiarized, including cases that went to court disputes and cases that stopped at the warning from the original author. In the case of melody plagiarism, as opposed to the entire melody being similar, there were many cases where the melody of certain sections was similar.

Cover song dataset was established using 10 pairs of original song of popular songs and cover songs sung by other people. As for the cover song, since each singer is different, the key of the song is different or slight melody modifications are included.

Plagiarism and Cover song datasets each contain 20 songs. Basically, it is an experiment to search pair song by comparing one query song with the remaining 19 songs. In order to verify whether it is possible to find pair song in the experiment, 81 songs at the test split of the custom dataset were randomly selected to be used additionally. In other words, except for the one song (i.e., the query object), 81 songs were additionally added to 19 songs to establish a setting of searching within 100 songs. The datasets consisting of the total of 101 songs are called Plagiarism 100

and Cover song 100, respectively.

### 7.2.2 Training

Model parameters were trained with Adam optimizer [83]. Learning rate was 0.0001, and if there was no improvement of validation loss during the 3 epoch, learning rate was reduced in half. If there was no performance improvement during 10 epochs, early stopping was applied to end training. Further, by selecting the epoch model with the best validation performance, it was utilized in the test. The batch size was 32.

### 7.2.3 Evaluation Metrics

For evaluation, metrics that are mainly used in cover song indentification were used. The metrics, such as, mean average precision (MAP), precision at 10 (P@10), and the mean rank of first correctly identified song (MR1) are metrics used in Mirex Audio Cover Song Identification contest[1]. Additionally, it was evaluated through the accuracy (Acc) which measures whether the pair song was searched first, and the median rank of first correctly identified song (MDR1).

[1] https://www.music-ir.org/mirex/wiki/2021:Audio_Cover_Song_Identification

## 7.3 Results

### 7.3.1 Quantitative Evaluation

Since we use self-supervised learning without a label, in order to evaluate the learned model, the performance on Plagiarism dataset and Cover song dataset was evaluated. And to analyze the factors that affect the model performance, we experimented by changing four factors with model structure, augmentation ratio, loss function, and definition of minimum distance between songs.

#### Model Structure

The experiment was conducted by changing the three model structures in turn. Melody is meaningful sequence data as an axis of time; therefore a model using LSTM [97], a recurrent neural network, was additionally utilized. Performance was evaluated using three LSTM alone, ResNet-based CNN, and ResNet-based CNN + LSTM. Table 7.1 shows the result of the performance by model structure from Plagiarism dataset. On the Plagiarism dataset, ResNet model showed the best performance, whereas when searching within Plagiarism 100, ResNet + LSTM's performance was better at MAP and Acc metric. However, when looking at the performance of P@10, MR1, and MDR1, ResNet model's performance was stable.

Table 7.2 shows the result of performance by model structure in the Cover song dataset. For all metrics, ResNet model's performance was the best. Overall, it seems that ResNet, which recognizes input as an image, is more appropriate than LSTM, which utilizes sequence information. Therefore,, in all other experiments, ResNet model was used.

Further, when comparing the performance between the Plagiarism dataset and

Cover song dataset, the latter showed a much better performance. In the case of plagiarism detection, even if the melody is slightly different, if people who hear it consider it similar, it can be plagiarism. Meanwhile, as for cover song identification, since it is a case where the same melody is sung by different people, it is a little easier to find.

Table 7.1: Performance on Plagiarism and Plagiarism 100 datasets according to the model structure.

| model | Results on Plagiarism | | | | | Results on Plagiarism 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| LSTM | 0.262 | 0.055 | 9.25 | 10 | 0.15 | 0.117 | 0.01 | 50.65 | 50 | 0.1 |
| ResNet | **0.48** | **0.085** | **5** | **3** | **0.3** | 0.225 | **0.05** | **23.15** | **10.5** | 0.15 |
| ResNet + LSTM | 0.399 | 0.09 | 5.5 | 5 | 0.25 | **0.236** | 0.03 | 29.35 | 26 | **0.2** |

Table 7.2: Performance on Cover song and Cover song 100 datasets according to the model structure.

| model | Results on Cover song | | | | | Results on Cover song 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| LSTM | 0.279 | 0.07 | 7.95 | 8 | 0.15 | 0.126 | 0.015 | 42.6 | 43.5 | 0.1 |
| ResNet | **0.784** | **0.09** | **3.3** | **1** | **0.75** | **0.56** | **0.08** | **14.45** | **2** | **0.45** |
| ResNet + LSTM | 0.573 | 0.085 | 4.7 | 2 | 0.45 | 0.3029 | 0.055 | 27.35 | 10 | 0.2 |

### Augmentation Ratio

The experiment was conducted using the value of minimum $r$ and maximum $R$ at the rate of augmentation used when creating a positive sample. By extracting a value randomly between $r$ and $R$, notes were selected to fit the ratio and transformed the notes. The higher the augmentation ratio, the more positive samples with more changes from anchors are created. The change of performance is as seen in Table 7.3 and Table 7.4.

On plagiarism detection, aside from the Plagiarism 100's Acc, the performance is optimal when the augmentation ratio is set between 0.3 and 0.4. On cover song identification, the small scale dataset showed a slightly good performance when the augmentation ratio was applied between 0.2 and 0.3. However, on Cover song 100 where the number of songs is large, the performance was superb when the augmentation ratio was applied between 0.3 and 0.4. When designing the loss function, positive sample was set to be farther than anchor and key-shifted anchor. It seems that the ratio to provide modification accordingly to the margin value was between 0.3 and 0.4. In order to remove the factors influencing the performance, other experiments used the min ratio at 0.2 and max ratio at 0.3.

Table 7.3: Performance on Plagiarism and Plagiarism 100 datasets according to augmentation ratio.

| min ratio (r) | Max ratio (R) | Results on Plagiarism | | | | | Results on Plagiarism 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| 0.1 | 0.2 | 0.384 | 0.07 | 6.9 | 5.5 | 0.25 | 0.158 | 0.04 | 34 | 27.5 | 0.1 |
| 0.2 | 0.3 | 0.48 | 0.085 | 5 | 3 | 0.3 | **0.225** | **0.05** | 23.15 | **10.5** | **0.15** |
| 0.3 | 0.4 | **0.539** | **0.09** | **4.4** | **2** | **0.4** | 0.224 | **0.05** | **21.4** | **10.5** | 0.1 |

Table 7.4: Performance on Cover song and Cover song 100 datasets according to augmentation ratio.

| min ratio (r) | Max ratio (R) | Results on Cover song | | | | | Results on Cover song 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| 0.1 | 0.2 | 0.523 | 0.075 | 5.95 | 2 | 0.4 | 0.405 | 0.06 | 22.95 | 5 | 0.3 |
| 0.2 | 0.3 | **0.784** | **0.09** | **3.3** | **1** | **0.75** | 0.56 | **0.08** | **14.45** | 2 | 0.45 |
| 0.3 | 0.4 | 0.766 | 0.85 | 3.6 | **1** | 0.7 | **0.687** | **0.08** | **14.45** | **1** | **0.6** |

**Loss Function**

In order to express the relations between key-shifted anchor and positive sample that are created using augmentation for self-supervised learning, the triplet loss function was applied to each of the three pairs. These loss functions are as shown in Equation 7.2, 7.3 and 7.4. In order to identify what impact each loss has on the model performance, evaluation was conducted on the performance depending on the use of three loss functions. The results are shown in Table 7.5 and Table 7.6. By default, the margin values $\alpha$, $\beta$ and $\gamma$ are set to 0.5, 1.0, and 1.5, respectively. When only using one out of the three loss functions, margin was adjusted to 1.0 for training.

First, comparing the case of using only one loss, the model trained with $Loss_1$ has the poorest performance. Since the case training with only $Loss_1$ does not utilize completely different negative samples, a positive sample with slight changes is recognized to be farther away. In such a case, it becomes impossible to distinguish between what is slightly different and completely different. When $Loss_2$ and $Loss_3$ were used independently, the performance overall seemed similar. $Loss_2$'s positive sample is a case with some modification at the anchor, as it includes key shift while negative sample is completely different; therefore, the model training was conducted somewhat normally. As for Loss 3, it is learned in the perspective that key-shifted anchor is a similar sample, while negative sample is completely different. As for plagiarism detection and cover song identification, in the perspective that we can detect as long as we can distinguish what is the same melody and not, training with $Loss_3$ alone can produce adequate model performance.

In the case using two loss functions, the performance of the model that trained

with $Loss_1$ and $Loss_3$ showed the poorest in both plagiarism detection and cover song identification. Due to the absence of $Loss_2$, which directly uses the relation between the positive sample and negative sample, it seems that distinguishing what is similar and what is not became difficult. As for the rest, the superiority of the performance changed depending on the dataset. The model which learned using all three loss functions showed the best performance at Plagiarism 100 dataset with P@10 and MDR1 metrics. As for Acc, although it is not the highest at Plagiarism 100 dataset, plagiarism detection seems to be somewhat stable. As for loss function, the procedure to find the optimum loss function through additional structure designing and hyperparameter adjustment is necessary. In other experiments, all three loss functions were applied.

Table 7.5: Performance on Plagiarism and Plagiarism 100 datasets according to loss function.

| $Loss_1$ | $Loss_2$ | $Loss_3$ | Results on Plagiarism | | | | | Results on Plagiarism 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| o | x | x | 0.359 | 0.065 | 7.35 | 7 | 0.25 | 0.122 | 0.02 | 38.6 | 36.5 | 0.05 |
| x | o | x | 0.481 | 0.085 | 5.35 | 4 | **0.35** | 0.275 | 0.045 | 26.35 | 21 | 0.2 |
| x | x | o | 0.507 | **0.09** | **4.35** | 3 | **0.35** | 0.242 | 0.045 | **19.85** | 12.5 | 0.15 |
| x | o | o | 0.469 | 0.075 | 5.95 | 4.5 | **0.35** | **0.322** | 0.04 | 30.2 | 22 | **0.25** |
| o | x | o | 0.287 | 0.06 | 7.55 | 7 | 0.15 | 0.094 | 0.02 | 32.7 | 27 | 0.05 |
| o | o | x | **0.512** | 0.085 | 5.4 | **2** | **0.35** | 0.19 | 0.04 | 26.45 | 12 | 0.05 |
| o | o | o | 0.48 | 0.085 | 5 | 3 | 0.3 | 0.225 | **0.05** | 23.15 | **10.5** | 0.15 |

Table 7.6: Performance on Cover song and Cover song 100 datasets according to loss function.

| $Loss_1$ | $Loss_2$ | $Loss_3$ | Results on Cover song | | | | | Results on Cover song 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| o | x | x | 0.271 | 0.08 | 7.1 | 6 | 0.1 | 0.114 | 0.025 | 32.45 | 22.5 | 0.05 |
| x | o | x | 0.813 | 0.08 | 4.05 | **1** | **0.8** | **0.803** | **0.08** | 14.35 | **1** | **0.8** |
| x | x | o | 0.795 | **0.095** | **3** | **1** | 0.75 | 0.779 | **0.08** | 11.8 | **1** | 0.75 |
| x | o | o | 0.731 | 0.08 | 4 | **1** | 0.65 | 0.646 | **0.08** | 13.1 | **1** | 0.6 |
| o | x | o | 0.332 | 0.095 | 4.95 | 5 | 0.1 | 0.163 | 0.04 | 19.1 | 15 | 0.05 |
| o | o | x | **0.82** | 0.09 | 3.05 | **1** | **0.8** | 0.78 | **0.08** | **10.85** | **1** | 0.75 |
| o | o | o | 0.784 | 0.09 | 3.3 | **1** | 0.75 | 0.56 | **0.08** | 14.45 | 2 | 0.45 |

### Definition of Minimum Distance Between Songs

By measuring the distance of all the pairs of query song and search object songs, the shortest distance was defined as the distance between the two songs. However, since this only considers the distance among the four bars, it does not take into account cases where the similar part continues longer than four bars. Such characteristic is more noticeable in cover song identification than plagiarism detection. While similar melody occurs in only certain parts for plagiarism, the cover song, on the other hand, has continued similar melodies. Therefore, it can be defined that the shorter the average distance of the continuing pair within each song, it is more similar. This can be expressed as below.

$$Distance(X, Y) = \min_{i,j} \sum_{k=0}^{K-1} \|f_{model}(x_{i+k}) - f_{model}(y_{j+k})\|_2^2 \tag{7.6}$$

$X$ and $Y$ are songs, while $f_{model}$ is the deep learning model. $K$ is the number of input considered together in order.

Table 7.7 organizes the performance by the change of $K$ value. By the standards of MAP with large deviation, on plagiarism 100, the best performance was shown when $K$ was 7. If $K$ becomes larger or smaller than that, performance drops. We can identify that finding similarity in somewhat continuing sections is helpful. The percentage of finding a plagiarized song accurately among the 100 songs is 40 percent. When looking at P@10, 50 percent found plagiarism songs in the top 10.

For Cover song 100, the performance was at best when $K$ was 18. We can identify the tendency that as $K$ increases, the value of MAP increases and when $K$ is 18, MAP and Acc values are the highest. The search performance in cover song identification improves when considered long in the order.

Table 7.7: Performance on Plagiarism 100 and Cover song 100 datasets according to the number of input considered together in order ($K$).

| K | Results on Plagiarism 100 | | | | | Results on Cover song 100 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MAP | P@10 | MR1 | MDR1 | Acc | MAP | P@10 | MR1 | MDR1 | Acc |
| 1 | 0.225 | 0.05 | **23.15** | 10.5 | 0.15 | 0.56 | **0.08** | 14.45 | 2 | 0.45 |
| 2 | 0.274 | 0.04 | 30.1 | 17 | 0.2 | 0.627 | **0.08** | 14.5 | 1.5 | 0.5 |
| 3 | 0.31 | 0.045 | 25.65 | 13 | 0.25 | 0.67 | **0.08** | 13.4 | **1** | 0.6 |
| 4 | 0.351 | 0.05 | 24.5 | 10 | 0.25 | 0.746 | **0.08** | 12.85 | **1** | 0.7 |
| 5 | 0.387 | **0.055** | 25.65 | **8** | 0.3 | 0.744 | **0.08** | 11.45 | **1** | 0.7 |
| 6 | 0.431 | 0.05 | 25.2 | 8.5 | 0.35 | 0.742 | **0.08** | 9.55 | **1** | 0.7 |
| 7 | **0.449** | 0.05 | 26.45 | 11.5 | **0.4** | 0.744 | **0.08** | 9.65 | **1** | 0.7 |
| 8 | 0.446 | 0.05 | 26.15 | 11 | **0.4** | 0.729 | **0.08** | 9.8 | **1** | 0.7 |
| 9 | 0.444 | 0.045 | 27.4 | 12.5 | **0.4** | 0.707 | **0.08** | 9.3 | **1** | 0.65 |
| 10 | 0.437 | 0.05 | 27.4 | 13 | **0.4** | 0.702 | **0.08** | 10 | **1** | 0.65 |
| 11 | 0.392 | 0.05 | 27.8 | 10.5 | 0.35 | 0.705 | **0.08** | 9.8 | **1** | 0.65 |
| 12 | 0.353 | 0.045 | 29.1 | 15.5 | 0.3 | 0.7 | 0.075 | 8.7 | **1** | 0.65 |
| 13 | 0.342 | 0.04 | 29.15 | 18 | 0.3 | 0.73 | 0.075 | **8.6** | **1** | 0.7 |
| 14 | 0.336 | 0.04 | 29.1 | 15 | 0.3 | 0.738 | 0.075 | 8.8 | **1** | 0.7 |
| 15 | 0.338 | 0.045 | 28.2 | 13.5 | 0.3 | 0.739 | **0.08** | 9.25 | **1** | 0.7 |
| 16 | 0.341 | 0.045 | 28.1 | 17 | 0.3 | 0.737 | **0.08** | 9.7 | **1** | 0.7 |
| 17 | 0.347 | 0.045 | 26.9 | 14 | 0.3 | 0.74 | **0.08** | 9.55 | **1** | 0.7 |
| 18 | 0.358 | 0.05 | 26.35 | 10.5 | 0.3 | **0.773** | **0.08** | 9.85 | **1** | **0.75** |
| 19 | 0.368 | 0.05 | 26.95 | 11.5 | 0.3 | 0.748 | **0.08** | 10 | **1** | 0.7 |
| 20 | 0.368 | 0.045 | 27.75 | 15.5 | 0.3 | 0.738 | **0.08** | 10.6 | **1** | 0.7 |

### 7.3.2 Qualitative Evaluation

**Plagiarism cases**

As for plagiarism pairs, there were both cases where the search was successful and it failed. In order to analyze the reason, we analyzed the search results. First, we intend to analyze the cases where the search of plagiarized song within 100 songs was successful as it showed as the first outcome. It is a case where a plagiarized song Bandido-Vamos Amigos[2] was found when searching query for the song Lee Jung Hyun-Wa[3]. It can be identified in Figure 7.5.

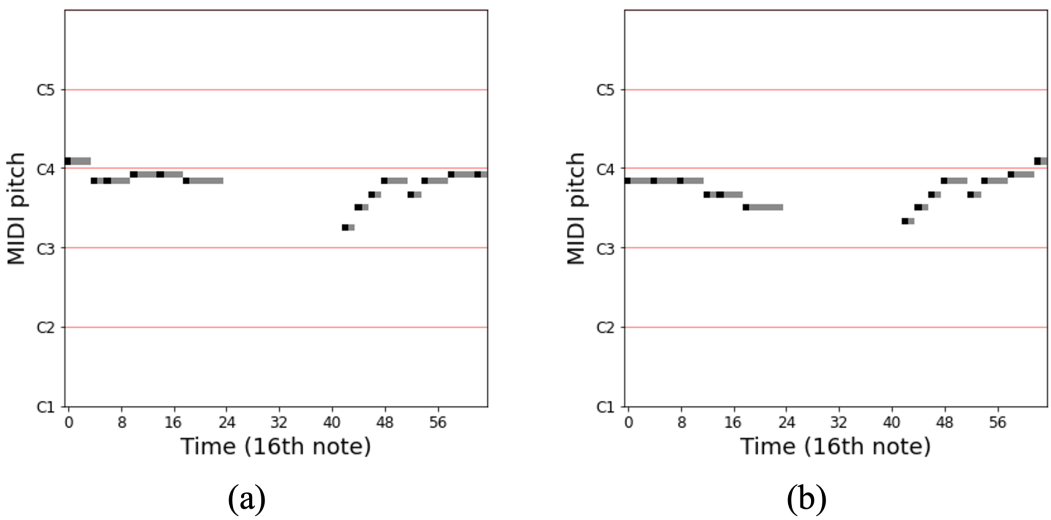



Figure 7.5: (a) is 54.1∼60.91 seconds of Lee Jung Hyun-Wa. (b) is 95.87∼102.69 seconds of Bandido-Vamos Amigos.

While some sections differ, the melody is very similar in general. In this case, the distance between the two sections was 0.17, being the closest section among the 100 songs. When expressing the distance of all the sections with respect to the two songs

[2]https://www.youtube.com/watch?v=e2HYsbUiwLk
[3]https://www.youtube.com/watch?v=ZblHv1Lpyfk

in similarity matrix, it is as Figure 7.6. The similar sections are expressed in dark color and you can identify a dark diagonal line. You can identify the similar sections continuing in the plagiarism case. In Section 7.3.1, when expressing the minimum distance between the songs as the sum of the continuing sections, we can identify why the detection performance improves. It can be seen that there are many cases of plagiarism when consecutive sections are similar.

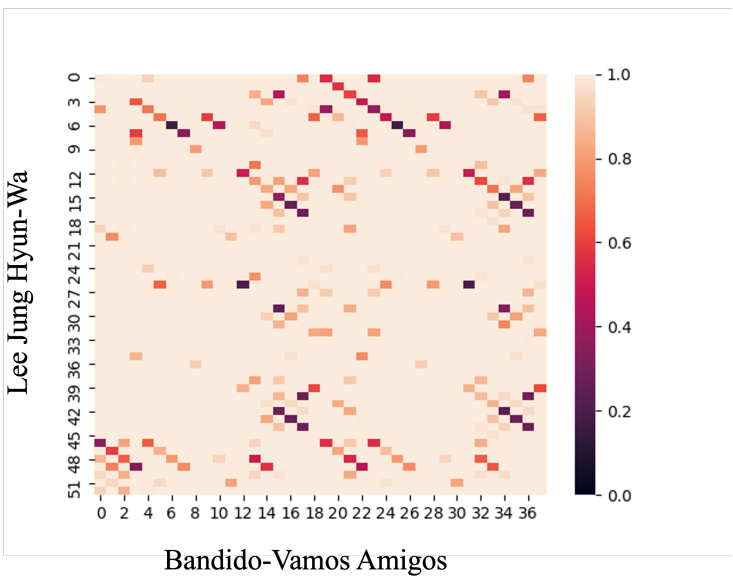



Figure 7.6: Lee Jung Hyun-Wa and Bandido-Vamos Amigos's all sections pairs' distances are calculated and expressed in similarity matrix.

We intend to analyze the second case where the search of plagiarism song was successful. It is a case of finding a plagiarized song called Seiell-Scenne Nenne[4] when searching the song BTS-Fake Love[5] in query. It is seen in Figure 7.7.

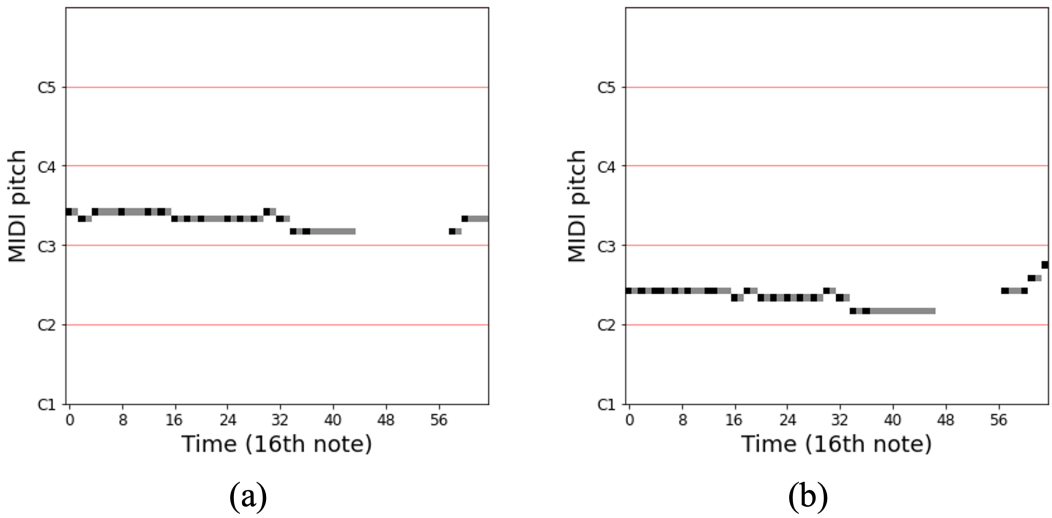



Figure 7.7: (a) is 62.89∼69.09 seconds of BTS-Fake Love. (b) is 97.43∼103.61 seconds of Seiell-Scenne Nenne.

The distance between the two sections was 0.13, searched as the closest section among the 100 songs. While the melody of (a) and (b) sections is very similar, key shift appeared very large. Despite, the two melodies are expressed similar in embedding space since the self-supervised learning utilizing the key-shifted anchor worked effectively.

Figure 7.8 shows the expression of the distance of all sections with respect to the two songs. The dark diagonal lines are seen in many places. Since the section that is plagiarism is repeated, there are many diagonal lines.

[4]https://www.youtube.com/watch?v=Akob0Smf9Ag
[5]https://www.youtube.com/watch?v=NT8ePWlgx_Y

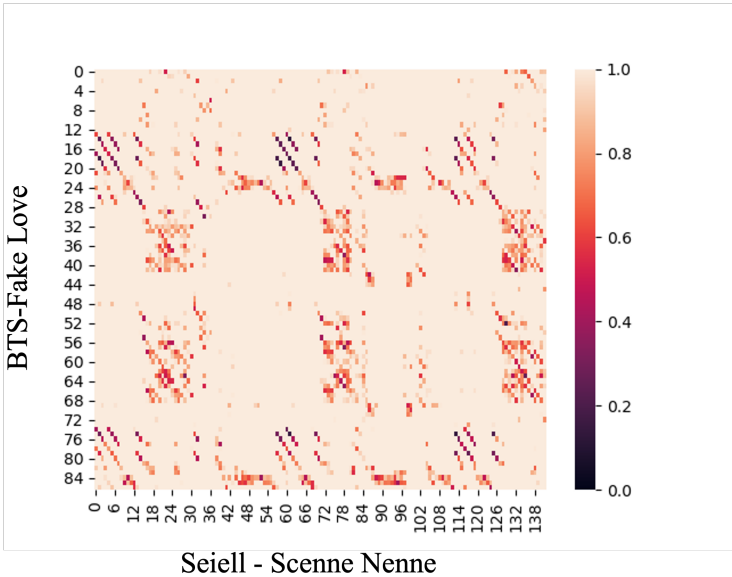



Figure 7.8: BTS-Fake Love and Seiell-Scenne Nenne's all sections pairs' distances are calculated and expressed in similarity matrix.

The failed search case involved being unable to discover Ed Sheeran-Shape of You[6] when searching TLC-No Scrubs[7] in the query. The minimum distance between the two songs was the low at being 0.12. However, since many other songs with lower distance were found, the song was searched as the 11th closet song among the 100 songs. Figure 7.9 represents the pair target and failure case.

[6]https://www.youtube.com/watch?v=JGwWNGJdvx8
[7]https://www.youtube.com/watch?v=FrLequ6dUdM

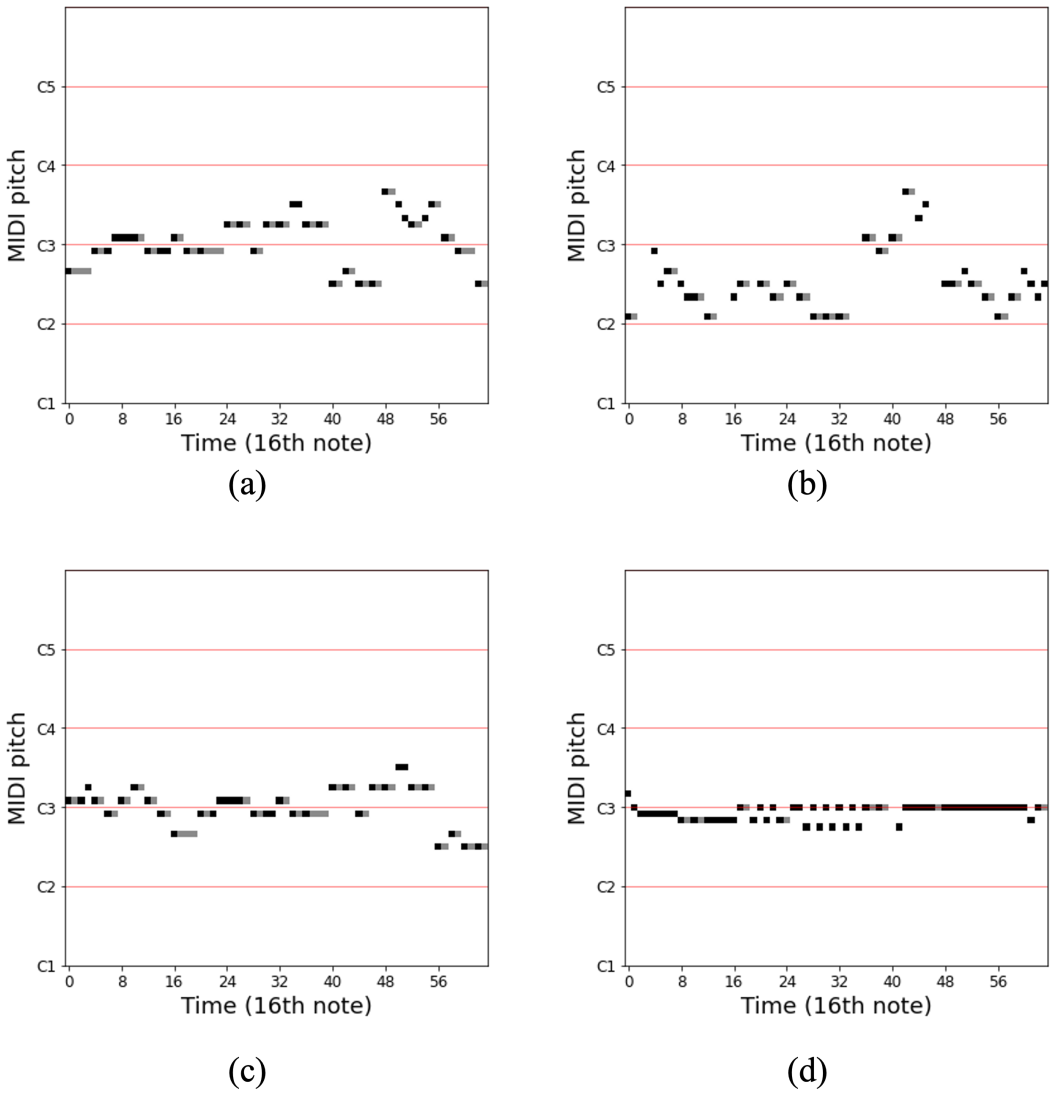



Figure 7.9: (a) is 55.41∼65.74 seconds of TLC-No Scrubs. (b) is 41.13∼51.13 seconds of Ed Sheeran – Shape of You. (c) is 52.83∼63.16 seconds of TLC-No Scrubs. (d) is a section of an unrelated song on the test.

While (a) and (b) are not very similar in terms of the image, they contain sections about plagiarism. The distance between the two sections is 0.12. (c) and (d) are completely unrelated songs and can be seen completely different in image. However, the distance between the two sections is mapped as very close, distance at 0.08. This shows the limitations of self-supervised learning where an unrelated melody was learned to be similar. In order to address this issue, the quality of generating training data using augmentation should be improved, and adequate loss function and model structure should be supported.

When looking at the failed cases of plagiarism detection, the issues are narrowed down to two. There are cases where the similar melody is recorded to be different due to the insufficient performance of melody transcription. In addition, as seen above, there are cases where completely different melodies are considered similar. This is a problem which occurs because music plagiarism detection is conducted in 2 phases, requiring improvement of performance respectively.

### Additional Similarity Matrix

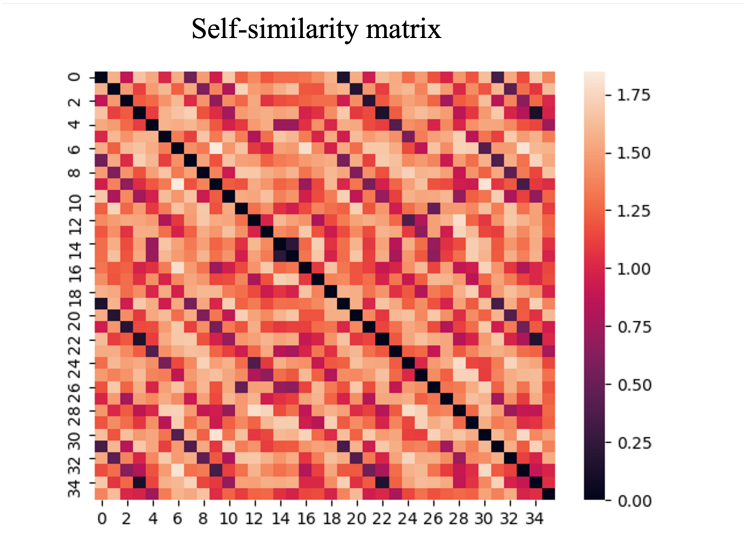



Figure 7.10: Distance for all sections of a song of test data is calculated and expressed in self-similarity matrix.

First, when drawing the self-similarity matrix on a song, it is as seen in Figure 7.10. This song has the characteristic that similar melodies repeat. This can be identified in the figure as many diagonal lines are seen aside from the diagonal line in the center. This indicates same melody sections repeat. Through this, we can also predict the repeating structure of the song. Through the analysis of similarity matrix, it seems possible to somewhat analyze the repeating structure of songs.

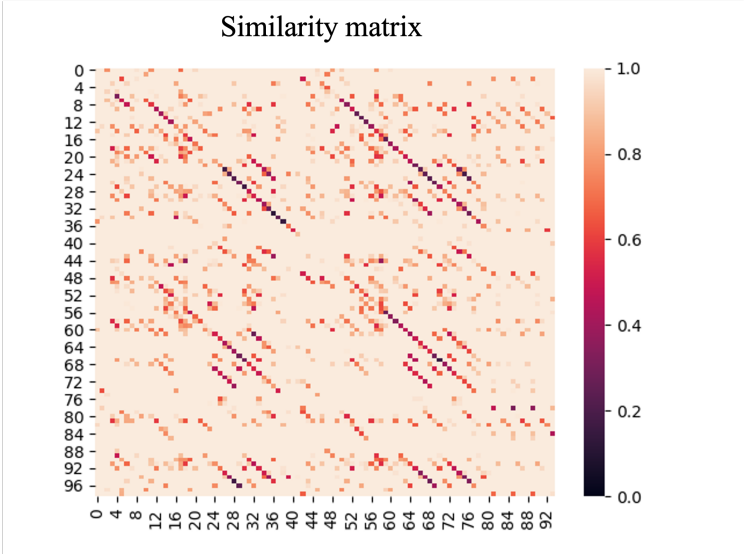



Figure 7.11: Distance for all sections of the original and cover song is calculated and expressed in similarity matrix.

The similarity matrix between the original song and cover song used in the cover song identification is shown in Figure 7.11. The section that is recognized to be similar is displayed as very long. A cover song is a song in which the same melody is sung in different voice over a long period of time. Therefore, while in the case of plagiarism songs, certain sections that are similar appear short, while over songs display the similar sections as long.

# Chapter 8

# Conclusion

## 8.1 Summary and Contributions

This thesis sought to transcribe automatic lead sheet from music audio signals and to study its application. For this, the research for the recognition of each of chord and melody were conducted and we proposed the lead sheet transcription methods by combing these. Further, by focusing on the melody among the recognized lead sheet, the thesis explored melody similarity assessment.

First, we presented bi-directional Transformer for chord recognition (BTC). The self-attention mechanism was appropriate for the task that attempts to capture long-term dependency by effectively exploring relevant sections. BTC has an advantage in that its training procedure is simple and it showed results competitive to other models in most of the evaluation metrics. Through the attention map analysis, it turned out that each self-attention layer had different characteristics and that the attention mechanism was effective in identifying sections of chords that were crucial for chord recognition.

We also proposed a monophonic note-level singing transcription model using a sequence-to-sequence Transformer that advances state-of-the-art singing transcription on MIR-ST500 dataset. Accordingly, we introduced a method of representing

monophonic melodies as musical event sequences and approached singing melody transcription through sequence-to-sequence task. The overlapping decoding turned out to be effective for note offset prediction by preserving sequential context information. The transcription performance was also improved by introducing pitch augmentation and adding noisy dataset with data cleansing, having effects in preventing overfitting and training a robust model. Visualization of the transcription results enabled qualitative analyses to investigate the effect of each of the proposed techniques. Subjective human evaluation showed that the results of our proposed model were perceived as more accurate than those of a previous study.

By combining the two preceding research results, this thesis proposed an automatic lead sheet transcription method. It utilized the previously researched key estimation and beat tracking to suggest a method to combine various information extracted from audio. The process for automatic lead sheet transcription consisting of various steps was introduced. And we analyzed transcription performance by comparing it with the expert's transcription.

The thesis explored melody similarity assessment as one of the methods of applying the lead sheet transcription technology. By focusing on the cases of melody, we suggested a ResNet model that converts melodies into embeddings. In order to train the model, self-supervised learning method utilizing musical data augmentation was also proposed. In addition, we introduced loss function that reflects the characteristics of melody similarity. The results of the experiment demonstrated the possibility of music plagiarism detection and cover song detection. And we identified further application potentials, such as finding repeated structure within a song.

Automatic music lead sheet transcription technology can offer great assistance to

various people in the music industry and instrument players. Through this, the time and cost consumed for transcription could be reduced. Combining and organizing the complicated lead sheet transcription process is the biggest contribution of the paper. Further, identifying the possibility of plagiarism detection at the level of sound sources is an important result. As far as we know, no research has yet shown proper plagiarism detection at the level of actual popular song sources. If such plagiarism detection becomes popular, it can have a good effect on composers in reducing cases of unintentional plagiarism.

## 8.2 Limitations and Future Research

It is a great merit that the automatic lead sheet transcription is possible at the actual sources of sound. However, there are many instances where singing melody and chord transcriptions alone are insufficient. Therefore, if there is a technology that allows the dissection and transcription of the sound of various musical instruments, it will benefit people even more. In addition, when it comes to singing melody transcription, polyphonic melody or chorus cannot be recognized. If this could also be perfectly recognized, the applicability will greatly increase. Through this, studies on source separation by vocals could be possible. Additionally, the fundamental solution to improve the performance of automatic music transcription is to collect sufficient training data. In the future, more automatic sheet music alignment studies need to be conducted to provide additional annotation data.

In the case of plagiarism, it has limitations that it can be utilized only for singing melody plagiarism. Aside from singing melody, there are diverse types of plagiarism such as instrument melody and sample plagiarism. For the study of instrument melody plagiarism, instrument melody transcription technique is essential, while for sample plagiarism, technology to apply right away at the level of audio signals is needed. Although this thesis introduced music plagiarism detection based on transcription, it is also possible to directly embed the audio signal and compare the distance. Applying an audio-based similarity evaluation model for music plagiarism detection is left for future work.

# Bibliography


[1] S. Kum and J. Nam. Joint detection and classification of singing voice melody using convolutional recurrent neural networks. *Applied Sciences*, 9(7):1324, 2019.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, pages 6000–6010, 2017.

[3] J. Devlin, MW. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] TB. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6706–6713, 2019.

[6] R. Hennequin, A.Khlif, F. Voituret, and M. Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020.

[7] P. Dhariwal, H. Jun, C. Payne, JW. Kim, A. Radford, and I. Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[8] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam. Disentangled multidimensional metric learning for music similarity. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2020.

[9] A. Sheh and D. P. W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, Baltimore, Maryland, USA, 2003.

[10] F. Korzeniowski and G. Widmer. Improved chord recognition by combining duration and harmonic language models. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 10–17, Paris, France, 2018.

[11] Y. Wu and W. Li. Automatic audio chord recognition with midi-trained deep feature and BLSTM-CRF sequence decoding model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):355–366, 2019.

[12] J. Y. Wang and J. S. R. Jang. On the preparation and validation of a large-scale dataset of singing transcription. In *IEEE International Conference on*

*Acoustics, Speech and Signal Processing (ICASSP)*, pages 276–280, Toronto, Canada,, 2021.

[13] J. Gardner, I. Simon, E. Manilow, C. Hawthorne, , and J. Engel. Mt3: Multi-task multitrack music transcription. *arXiv preprint arXiv:2111.03017*, 2021.

[14] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii. Audio-to-score singing transcription based on a crnn-hsmm hybrid model. *APSIPA Transactions on Signal and Information Processing*, 10:e7, 2021.

[15] C. Donahue and P. Liang. Sheet sage: Lead sheets from music audio. In *Extended Abstracts for the Late-Breaking Demo Session of the International Society for Music Information Retrieval Conference*, Online, 2021.

[16] R. D. Prisco, A. Esposito, N. Lettieri, D. Malandrino, D. Pirozzi, G. Zaccagnino, and R. Zaccagnino. Music plagiarism at a glance: Metrics of similarity and visualizations. In *2017 21st International Conference Information Visualisation (IV)*, pages 410–415, 2017.

[17] C. Dittmar, K. F. Hildebrand, D. Gaertner, M. Winges, F. Müller, and P. Aichroth. Audio forensics meets music information retrieval — a toolbox for inspection of music plagiarism. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1249–1253, 2012.

[18] R. D. Prisco, D. Malandrino, G. Zaccagnino, and R. Zaccagnino. Fuzzy vectorial-based similarity detection of music plagiarism. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2017.

[19] T. He, W. Liu, C. Gong, J. Yan, and N. Zhang. Music plagiarism detection via bipartite graph matching. *arXiv preprint arXiv:2107.09889*, 2021.

[20] K. Park, S. Baek, J. Jeon, and Y.-S. Jeong. Music plagiarism detection based on siamese cnn. *HUMAN-CENTRIC COMPUTING AND INFORMATION SCIENCES*, 12, 2022.

[21] M.-S. Sie, C.-C. Chiang, H.-C. Yang, and Y.-L. Liu. A novel method of plagiarism detection for music melodies. *International Journal of Artificial Intelligence and Applications*, 8:15–31, 09 2017.

[22] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park. A bi-directional transformer for musical chord recognition. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 620–627, 2019.

[23] J. Park, K. Choi, S. Oh, L. Kim, and J. Park. Note-level singing melody transcription with transformers. *Intelligent Data Analysis*, forthcoming.

[24] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[25] CZ. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck. Music transformer: Generating music with long-term structure. *arXiv preprint arXiv:1809.04281*, 2018.

[26] K. Choi, J. Park, W. Heo, S. Jeon, and J. Park. Chord conditioned melody generation with transformer based decoders. *IEEE Access*, 9:42071–42080, 2021.

[27] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel. Sequence-to-sequence piano transcription with transformers. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 246–253, Online, 2021.

[28] L. Euler. *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae.* ex typographia Academiae scientiarum, 1739.

[29] T. Cho. *Improved Techniques for Automatic Chord Recognition from Music Audio Signals.* PhD thesis, New York University, 2014.

[30] T. Cho and J. P. Bello. A feature smoothing method for chord recognition using recurrence plots. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 651–656, Miami, Florida, USA, 2011.

[31] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama. Hmm-based approach for automatic chord detection using refined acoustic features. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5518–5521, Dallas, Texas, USA, 2010.

[32] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *11th International Conference on Machine Learning and Applications (ICMLA)*, pages 357–362, Boca Raton, FL, USA, 2012.

[33] E. J. Humphrey, T. Cho, and J. P. Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *IEEE International Conference*

*on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456, Kyoto, Japan, 2012.

[34] F. Korzeniowski and G. Widmer. Feature learning for chord recognition: The deep chroma extractor. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 37–43, New York City, USA, 2016.

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

[36] X. Zhou and A. Lerch. Chord detection using deep learning. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 52–58, Málaga, Spain, 2015.

[37] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 335–340, Curitiba, Brazil, 2013.

[38] S.Sigtia, N. Boulanger-Lewandowski, and S.Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 127–133, Málaga, Spain, 2015.

[39] F. Korzeniowski, D. R. W. Sears, and G. Widmer. A large-scale study of language models for chord prediction. In *IEEE International Conference on*

*Acoustics, Speech and Signal Processing (ICASSP)*, pages 91–95, Calgary, AB, Canada, 2018.

[40] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR)*, 2015.

[41] M. Mauch and S. Dixon. Pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, 2014.

[42] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[43] Y.-T. Wu, Y.-J. Luo, T.-P. Chen, I-C. Wei, J.-Y. Hsu, Y.-C. Chuang, and L. Su. Omnizart: A general toolbox for automatic music transcription. *arXiv preprint arXiv:2106.00497*, 2021.

[44] L. Su. Vocal melody extraction using patch-based cnn. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 371–375, 2018.

[45] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019.

[46] T. Miyato, S. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE*

*Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

[47] M. Tan and Q. V. LE. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[48] S. Kum, J. Lee, K. L. Kim, T. Kim, and J. Nam. Pseudo-label transfer from frame-level to note-level in a teacher-student framework for singing transcription from polyphonic music. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 796–800, 2022.

[49] S. Kum, J. H. Lin, L. Su, and J. Nam. Semi-supervised learning using teacher-student models for vocal melody extraction. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 93–100, Montréal, Canada, 2020.

[50] K. Noland and M. Sandler. signal processing parameters for tonality estimation. *journal of the audio engineering society*, 2007.

[51] S. Pauws. Musical key extraction from audio. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, Barcelona, Spain, 2004.

[52] Á. Faraldo, E. Gómez, S. Jordà, and P. Herrera. Key estimation in electronic dance music. In *Advances in Information Retrieval*, pages 335–347, Cham, 2016. Springer International Publishing.

[53] F. Korzeniowski and G. Widmer. End-to-end musical key estimation using a convolutional neural network. In *25th European Signal Processing Conference (EUSIPCO)*, pages 966–970. IEEE, 2017.

[54] F. Korzeniowski and G. Widmer. Genre-agnostic key classification with convolutional neural networks. *arXiv preprint arXiv:1808.05340*, 2018.

[55] H. Schreiber and M. Müller. Musical tempo and key estimation using convolutional neural networks with directional filters. *arXiv preprint arXiv:1903.10839*, 2019.

[56] Y. Wu and K. Yoshii. Joint chord and key estimation based on a hierarchical variational autoencoder with multi-task learning. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[57] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer music journal*, 29(1):34–54, 2005.

[58] S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-11)*, pages 135–139, 2011.

[59] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 4, pages 2047–2052, 2005.

[60] S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 255–261, 2016.

[61] S. Böck, M. E.P. Davies, and P. Knees. Multi-task learning of tempo and beat: Learning one to improve the other. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 486–493, 2019.

[62] E. P. MatthewDavies and S. Böck. Temporal convolutional networks for musical audio beat tracking. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2019.

[63] P. Meier, G. Krump, and M. Müller. A real-time beat tracking system based on predominant local pulse information. In *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[64] M. Heydari and Z. Duan. Don't look back: An online beat tracking method using rnn and enhanced particle filtering. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 236–240, 2021.

[65] N. Borkar, S. Patre, R. S. Khalsa, R. Kawale, and P. Chakurkar. Music plagiarism detection using audio fingerprinting and segment matching. In *2021 Smart Technologies, Communication and Robotics (STCR)*, pages 1–4, 2021.

[66] Z. Yu, X. Xu, X. Chen, and D. Yang. Learning a representation for cover song identification using convolutional neural network. In *IEEE International*

*Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 541–545. IEEE, 2020.

[67] X. Du, Z. Yu, B. Zhu, X. Chen, and Z. Ma. Bytecover: Cover song identification via multi-loss training. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 551–555. IEEE, 2021.

[68] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[69] M. Arcos. Claraprint: a chord and melody based fingerprint for western classical music cover detection. *arXiv preprint arXiv:2009.10128*, 2020.

[70] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[71] K. Lee. Identifying cover songs from audio using harmonic representation. *MIREX 2006*, pages 36–38, 2006.

[72] J. P. Bello. Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 239–244, Vienna, Austria, 2007.

[73] J. Pauwels, F. Kaiser, and G. Peeters. Combining harmony-based and novelty-based approaches for structural segmentation. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 601–606, Curitiba, Brazil, 2013.

[74] T. Cho and J. P. Bello. On the relative importance of individual components of chord recognition systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(2):477–492, 2014.

[75] L. J. Ba, R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[77] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[78] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[79] B. Di Giorgi, M. Zanoni, A. Sarti, and S. Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, Erlangen, Germany, 2013.

[80] F. Korzeniowski and G. Widmer. A fully convolutional deep auditory model for musical chord recognition. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Vietri sul Mare, Salerno, Italy, 2016.

[81] B. McFee and J. P. Bello. Structured training for large-vocabulary chord recognition. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 188–194, Suzhou, China, 2017.

[82] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 367–372, Taipei, Taiwan, 2014.

[83] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[84] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[85] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[86] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[87] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.00497*, 2016.

[88] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu. On layer normalization in the transformer architecture. In *Proceedings of International Conference on Machine Learning (ICML)*, page PMLR 119, Online, 2020.

[89] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[90] R. L. Aguiar, Y. M.G. Costa, and C. N. Silla. Exploring data augmentation to improve music genre classification with convnets. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[91] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters. Dali: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 431–437, Paris, France, 2018.

[92] G. Meseguer-Brocal, R. Bittner, S. Durand, and B. Brost. Data cleansing with contrastive learning for vocal note event annotations. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 255–262, Montréal, Canada, 2020.

[93] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 155–160, 2014.

[94] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello. Deep salience representations for f0 estimation in polyphonic music. In *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, pages 63–70, Suzhou, China, 2017.

[95] Amazon. Amazon mechanical turk. `https://www.mturk.com`, 2022.

[96] C. Thomé, S. Piwell, and O. Utterbäck. Musical audio similarity with self-supervised convolutional neural networks. *arXiv preprint arXiv:2202.02112*, 2022.

[97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

# 국문초록

음악 산업의 디지털화를 통해 음악의 작곡, 편곡 및 유통이 편리해졌기 때문에 새롭게 공급되는 음원의 수가 증가하고 있다. 최근에는 누구나 크리에이터가 될 수 있는 플랫폼 환경이 구축되어, 사용자가 만든 자작곡, 커버곡, 리믹스 등이 유튜브, 틱톡을 통해 유통되고 있다. 이렇게 많은 양의 음악에 대해, 음악을 악보로 채보하고자 하는 수요는 음악가들에게 항상 존재했다. 그러나 악보 채보에는 음악적 지식이 필요하고, 시간과 비용이 많이 소요된다는 문제점이 있다.

본 논문에서는 심층 신경망을 활용하여 음악 리드 시트 악보 자동 채보 기법을 연구한다. 채보 인공지능의 개발은 음악 종사자 및 연주자들이 악보를 구하거나 만들기 위해 소모하는 시간과 비용을 크게 줄여 줄 수 있다. 또한 음원에서 디지털 악보 형태로 변환이 가능해지므로, 자동 표절 탐지, 작곡 인공지능 학습 등 다양하게 활용이 가능하다.

리드 시트 채보를 위해, 먼저 오디오 신호로부터 코드를 인식하는 모델을 제안한다. 음악에서 코드는 함축적이고 표현적인 음악의 중요한 특징이므로 이를 인식하는 것은 매우 중요하다. 코드 구간 인식을 위해, 어텐션 매커니즘을 이용하는 트랜스포머 기반 모델을 제시한다. 어텐션 지도 분석을 통해, 어텐션이 실제로 어떻게 적용되는지 시각화하고, 모델이 코드의 구간을 나누고 인식하는 과정을 살펴본다.

그리고 시퀀스 투 시퀀스 트랜스포머를 이용한 음표 수준의 가창 멜로디 채보 모델을 제안한다. 디코딩 과정에서 각 구간 사이의 문맥 정보가 단절되는 문제를 해결하기 위해 중첩 디코딩을 도입한다. 데이터 변형 기법으로 음높이 변형을 적용하는 방법과 데이터 클렌징을 통해 학습 데이터를 추가하는 방법을 소개한다. 정량 및 정성적인 비교를 통해 제안한 기법들이 성능 개선에 도움이 되는 것을 확인하였고, 제안모델이

MIR-ST500 데이터 셋에 대한 음표 수준의 가창 멜로디 채보 성능에서 가장 우수한 성능을 보였다. 추가로 주관적인 사람의 평가에서 제안 모델의 채보 결과가 이전 모델보다 저 정확하다고 인식됨을 확인하였다.

앞의 연구의 결과를 활용하여, 음악 리드 시트 자동 채보의 전체 과정을 제시한다. 오디오 신호로부터 인식한 다양한 음악 정보를 종합하여, 대중 음악 오디오 신호의 핵심을 표현하는 리드 시트 악보 채보가 가능함을 보인다. 그리고 이를 전문가가 제작한 리드시트와 비교하여 분석한다.

마지막으로 리드 시트 악보 자동 채보 기법을 응용하여, 자기 지도 학습 기반 멜로디 유사도 평가 방법을 제안한다. 리드 시트 채보 결과의 멜로디를 임베딩 공간에 표현하는 합성곱 신경망 모델을 제시한다. 자기지도 학습 방법론을 적용하기 위해, 음악적 데이터 변형 기법을 적용하여 학습 데이터를 생성하는 방법을 제안한다. 그리고 준비된 학습 데이터를 활용하는 심층 거리 학습 손실함수를 설계한다. 실험 결과 분석을 통해, 제안 모델이 표절 및 커버송 케이스에서 대중음악의 유사한 멜로디를 탐지할 수 있음을 확인한다.

**주요어**: 음악 정보 검색, 음악 자동 채보, 화음 인식, 가창 멜로디 인식, 멜로디 유사도 평가, 음악 표절 탐지, 자기지도 학습, 심층신경망

**학번**: 2018-20381