Doctoral Dissertation

Unsupervised Recognition of Chords, Functions, and Tonality

Yui Uehara

Supervisor: Satoshi Tojo

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
[Information Science]

September 2022

Abstract

Music has provided indispensable pleasures to humans and is one of the large markets in entertainment. Regularities that govern music have been studied for philosophical interests in ancient times and practical uses in the present age. As a result, music theories have converged into a certain degree of common sense: harmony theory. The harmony theory has been employed in music education and recent artificial intelligence. Despite its popularity, composers and even listeners have not fully been satisfied with it, and thus musical works have not been restricted by the theory.

The motivation of this study is to conduct a harmony analysis that reflects the characteristics of diverse musical expressions. Harmony analysis can be generalized as the following four processes. Firstly, define an appropriate set of chord labels. Then, segments scores and assign chord labels. Finally, analyze the labeled chord sequence by the chord functions. These processes seem simple but not trivial in practice because of their interdependency. It is especially significant for polyphonic music in contrast to music where melody and harmony can be easily distinguished (e.g., homophonic music). In addition, the notion of tonality would influence all these processes.

Previous efforts of unsupervised statistical learning for harmony have independently simulated chord labeling, chord function identification, or key detection. This study argues that simulations for these three attributes should be performed in a unified manner, considering the mutual dependency between them. In addition, chords and keys may not be easily annotated when we analyze a broader style of musical pieces. Therefore, we propose a model that does not require pre-annotations for not-targeted attributes and analyzes chord, function, and tonality in unified unsupervised learning.

To this end, this study attempts to combine a probabilistic generative model and neural networks. As the generative model, we select the hidden semi-Markov model (HSMM), an extension of the hidden Markov model (HMM). The HMM has been employed in previous works and showed promising results that well simulated known chord functions. However, considering that this study aims to automatically segment scores and classify chords instead of relying on pre-annotated chord symbols, we employ an HSMM that explicitly models duration probabilities for hidden states that are expected to represent latent chord categories. Furthermore, a more difficult problem arises in considering interdependencies between chord functions and tonality; chord functions, which is a notion that represents chord transition properties, are changed by local modulations, as H. Riemann pointed out.

In other words, a single transition matrix in conventional H(S)MM is no longer sufficient to analyze chord progressions when considering local modulations. Therefore, this study employs the idea of the neural hidden Markov model, which can adjust the hidden state transition probability by the contexts, and extends it to the semi-Markov model. The neural networks can utilize additional contexts, such as preceding chord sequences, pitch-classes, and beat information, for calculating categorical distributions that comprise the HSMM. Experiments show the added contexts considerably improve the model's generalization performance in terms of perplexity.

According to the aforementioned H. Riemann's view, tonality can be recognized by analyzing chord transition properties. We further introduce a teacher-student architecture to classify tonalities. While the teacher model equips the elaborated neural network for calculating the transition probability, the student model simplifies it to learnable matrices like a conventional HSMM. We prepare multiple student transition matrices and expect them to represent prototypes of tonalities. The student model classifies a predicted (labeled) chord sequence into a tonality by comparing the count of chord transitions with the transition probability matrices. Experiments show that the three-students model is the most consistent with the human analysis in terms of the F1-score; obtained three students can be interpreted as major, minor, and dorian modes, respectively. The transition matrices of the model reflect the difference between tonalities, consistently with known functions of tonic, dominant, and subdominant.

Thus, the neural HSMM and the extension of teacher-student architecture enable an unsupervised machine to recognize chords, chord functions, and tonality. We consult J. S. Bach's four-part chorales as the corpus of this study and give qualitative analysis in comparison with the conventional harmony theory. The consistency between the self-emergent chord functions and the known harmony theory suggests the potential of the proposed model to apply to a wider variety of music styles.

Keywords: Unsupervised Learning; Hidden Semi-Markov Model; Neural Network; Music Harmony Analysis; Chord Segmentation; Chord Function Recognition; Tonality Recognition.

Acknowledgments

Firstly, I express my sincere thanks to Professor Satoshi Tojo of Japan Advanced Institute of Science and Technology (JAIST), who has guided me throughout this research. His academic and musical knowledge have fostered my simple idea to a worthwhile study.

I wish to thank Professor Ryuhei Uehara of JAIST for his advice on my minor research project. I received valuable insights from the experience of the minor research. I also would like to thank Professor Masashi Unoki of JAIST, who gave me valuable comments that improved my thesis.

I am very grateful to Dr. Tetsuro Kitahara, Professor of Nihon University, for his precious advice for my research and academic knowledge of music informatics. I also wish to thank Dr. Daichi Mochihashi, Associate Professor of The Institute of Statistical Mathematics, for his valuable advice and professional knowledge for statistical natural language processing.

I am very grateful to my fellow students in the Tojo laboratory, Hiroyuki Yamamoto and Shuichi Shimosaka. I received a lot of inspiration from their passion and knowledge for music and technology.

I would also like to thank people who have influenced and supported me from before my Ph.D. study started. The knowledge and experiences I have received from musical professionals have been an essential basis for this study. Lastly, many thanks to my family for their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Composing or listening to music is one of the interesting abilities proper to human beings. Musical pieces are noticeably structured like human language, and thus their innate regularity allows us to understand and enjoy both the latest popular songs and ones made in the distant past. The challenge of finding the regularities in music have attracted musicologists for a long time ago [23, 69, 70, 74, 75]. The essence of their findings has been summarized in practical textbooks [3, 59, 67], which have been employed in music education and also artificial intelligence for music, such as structural analysis [34, 72], recommendation [53], melody harmonization [35, 81], and music generation [4, 61]. In reality, however, music is diverse and ambiguous. Thus, it is not easy to manually construct music grammars that organize the regularities and reflect the unique characteristics of individual pieces as well. Therefore, textbooks have avoided becoming overly complicated by basing the core parts of the theories on the most stylized pieces, such as Western classical music on the classical era and the Jazz standards [3, 59, 67].

Considering that these textbook theories have been derived from the experience of existing music, the question naturally arises, "Can a machine induce inherent regularities in music by statistical learning?". This study aims to identify hidden regularities in music; it is conceptually and technically relevant to grammar induction[1] in Natural Language Processing (NLP) [11, 12, 17, 18, 80]. In these studies, a statistical model is trained on data to obtain grammar for a language, but the same model can be trained on different data to learn grammar for a different language. In other words, the learned

[1] More specially, the techniques of this study are closely related to part-of-speech (POS) tag induction.

parameterizations for the statistical model represent the differences between languages. Utilizing this data-driven framework in music is expected to unify the view of musical phenomena while simultaneously revealing the individual characteristics of pieces.

The attempt to find latent regularities in music differs from more popular applications of music generation systems, while there are technical overlaps. Although the machines have acquired some abilities to create musical pieces by employing state-of-the-art techniques such as Transformers [43], the mechanism is intuitively hard for humans to understand and interact with the system. Since elaborations of surface pitch events vary depending on genre or instrumentation, humans need more robust and structured knowledge to be utilized in creative works. Because of this reason, musicology has described the nature of music in more abstract units such as chords and functions. Therefore, this research performs feature acquisition for the known concepts in musicology: chord, function and tonality.

Compared to direct applications such as music generation systems, data-oriented statistical acquisition of regularity in harmony has been rarely investigated [44, 73, 82, 88]. They have found similarities between statistically induced clusters and chord functions in textbook theories under the limited condition where chord symbols, segmentation, and keys are given [44, 82, 88]. However, the concepts of chords, chord functions, and tonality[2] are mutually dependent in reality[3]. Therefore, this dissertation tries to analyze these three features in a unified model. Since only a limited number of pieces can be easily pre-processed for chords and keys, the proposal is essential in developing a data-driven harmony analysis for various pieces in the future.

1.2 Research Objective

This study aims to recognize chord, function, and tonality based on unsupervised learning. It is not an implementation of textbook knowledge; instead, a proposed statistical model is tuned with raw data of musical pieces. Although our future goal is that a statistical model extensively recognizes the characteristics and innate regularities of diverse masterpieces, we start by experimenting with our model focusing on J. S. Bach's four-part chorales; and then examine the self-emergent patterns.

This study employs data-driven, unsupervised learning; however, it con-

[2]We use the term "tonality" in a somewhat broader meaning that indicates basically a key but sometimes a *church mode* that is employed within tonal music. However, we use "key" instead when targeting only modern 24 keys.

[3]More detail will be described in Chapter 2.

ceptually presupposes notions of musical knowledge such as chord, chord function, and tonality. We regard these notions as corresponding to the following statistical learning.

1. Chords are recognized by automatic segmentation and categorization of surface pitch events.

2. Chord functions are statistical properties of transitions between chord categories.

3. Tonality or key recognition is a categorization of chord transitions.

Here, we also take over a popular assumption of transpositional equivalency between keys, which is employed in general music theories and previous studies [42, 44, 51, 73, 75, 88]. We do not consider the absolute position of the main tonality but only the relative relationship between the main tonality and local modulations. Therefore, targeted pieces are transposed so as not to have no key signature before the analysis; this is automatically processed from the key signature. Note that we neither apply modulation segmentation nor mode(major or minor, and possibly church modes) classification that are not self-evident only by a score. In classical music, in particular, we can observe a wide variety of modulation techniques and developments, even if the main key is normalized.

This study follows previous studies that targeted the statistical classification of chord functions [44, 73, 82, 88]; however, it attempts to remove their limitations that chord labeling/segmentation, mode classification, and modulation segmentation were given in advance. We analyze these features with a unified model since chord, function, and tonality are mutually dependent and should not be analyzed independently.

We select the hidden semi-Markov model (HSMM) as the model for this study, the structure of which well agrees with the above concepts of chords and chord functions. HSMM equips three components of categorical distributions: hidden state transition, hidden state duration, and emission. The model is suitable for automatic segmentation and categorization for chords; the hidden states represent chord categories, and the duration probability predicts the lengths of chord categories. Unlike hidden Markov model (HMM), HSMM avoids the useless increase of the ratio of self-transition by providing duration probability for the chord length; this allows the hidden state transition probability to reflect the regularity of transitions between chord categories as data-oriented chord functions.

However, the conventional HSMM would not be robust enough to obtain appropriate chords and chord functions from the complicated surface pitch

events and local modulations. Therefore, we employ the idea of neural hidden Markov model (HMM) and extend it to HSMM. The strength of utilizing neural networks is that they allow for integrating additional features such as metrical information and preceding observations. Experiments show the effectiveness of additional contexts in terms of perplexity and quality for induced chord categories.

We expect that an appropriate set of chord categories and tonalities (a set of closely related keys appeared in modulations) vary between targeted music. The proposed model's objective is to produce an adequate interpretation of chords, chord functions, and tonalities, with self-emergent categories, just by raw scores [4] as input.

1.3 Dissertation Outline

We have introduced the motivation behind this work and presented the research objective. The rest of this dissertation is organized as follows.

We first give some examples of musical works to show the efficacy and limitations of the conventional harmony theory in Chapter 2. The first example in the chapter (Section 2.1) also serves as a guide for the theory. On the other hand, we show how a masterpiece is beyond the stereotypical patterns in Section 2.2. Then, we give a further example (Section 2.3) that is hardly analyzed by the conventional harmony theory even though a sense of tonality is retained.

In Chapter 3, we introduce related works of statistical learning for music (Section 3.1) and give technical backgrounds (Section 3.3). In addition, we describe the background of J. S. Bach's four-part chorales dataset that is used as the corpus for this study in Section 3.2.

The following two chapters construct the central part of our study. In Chapter 4, we propose a model that automatically segments surface pitch events without given pre-defined chord symbols and key annotations. In order to achieve this, we employ hidden semi-Markov model (HSMM) and incorporate the techniques of the neural HMM. In Chapter 5, the model is extended to classify chord sequences into tonalities by introducing a *teacher-student* architecture.

Finally, we conclude the achievements of this study and discuss the future directions in Chapter 6.

[4] We assume scores to be machine-readable, in particular, with MusicXML format.

Chapter 2

Harmony Theory

This chapter gives three examples of Western classical music to explain the efficacy and limitation of conventional harmony theory. The first one (Section 2.1) is an example well described by the theory. In the second example (Section 2.2), we can see a sophisticated technique of how a composer avoids stereotypical chord progressions to create impressive work. Finally, we show that the conventional theory is almost helpless to describe the example of Section 2.3, even though it retains a sense of tonality. In other words, it has a sense of tonal center and closely related keys where some extent of the regularity of chord progressions can be heard. The question of how we can make models to discover such hidden regularities is the starting motivation of our study.

2.1 An Exemplary Chord Progression

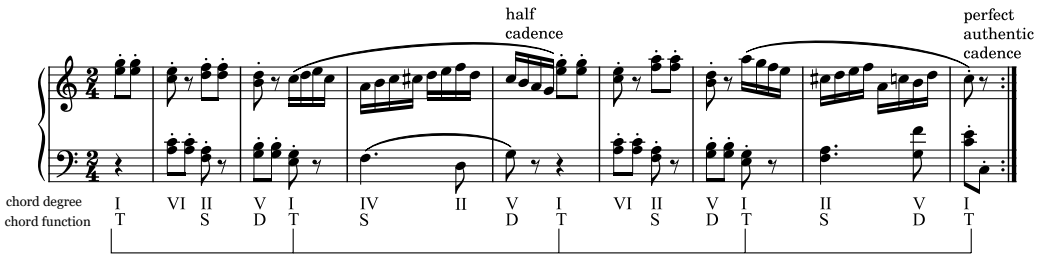


Figure 2.1: W. A. Mozart: Piano sonata No.16, beginning part of 3rd movement.

An exemplary chord progression can be found in a piano sonata by W. A. Mozart (**Figure 2.1**). Most theories employ the notion of *chord functions* [70]: tonic (T), dominant (D), and subdominant (S). A Roman numeral

represents a *chord degree*, which corresponds to the degree between the *root note* of a chord and that of the tonic chord (**I**). Correspondences between chord degrees and chord functions are as follows: tonic = { **I**, **III**, **VI** }, dominant = { **V**, **VII** }, and subdominant = { **II**, **IV** }.

Surprisingly, the chord functions that consist of only three types of T, D, and S describe typical chord progressions quite well. A unit of chord progressions is a chunk that starts with T and ends with T. Basically, variations of units are only three types:

- T → D → T
- T → S → D → T
- T → S → T

Note that the progression of D → S is assumed to be rare, and thus not found above. In fact, the example of **Figure 2.1** consists of four chunks of T → S → D → T.

2.2 Techniques Beyond the Stereotype

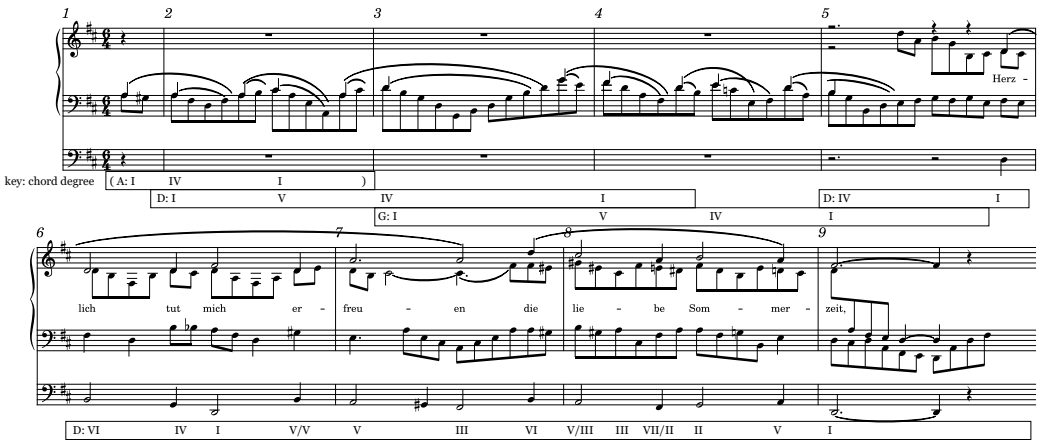


Figure 2.2: J. Brahms: Op. 122 No.4

We have seen that the chord functions successfully described Mozart's piano sonata. However, it would be too simplified to represent the rich diversity of musical pieces, which have been highly elaborated to be unique as art.

A good example is the beginning part of an organ chorale by J. Brahms (**Figure 2.2**). In the beginning, while the key signature indicates **D** major,

the first three notes A G♯ A induce the expectation of an **A** major key. This expectation is failed at bar 3 by the G major chord. Usually, when a key changes to another (modulation), the two keys share at least one chord. [75]. Therefore, listeners would modify the interpretation to **D** major key and consider the possibility of **G** major key for the next. However, the strange thing is that the chord progression is dominant (**V**) → subdominant (**IV**) at bars 2 to 3, which is unusual as described above. The expectation of **D** major key failed again at bar 4 by the C major chord. Here, the S → D progression appears again with **G** major key. Since the dominant strongly expects tonic, this progression and modulations on the subdominant circle (**A** major → **D** major → **G** major) would make listeners feel unanswered or like floating. Combined with the light texture, this opening of the piece sounds like a gentle breeze.

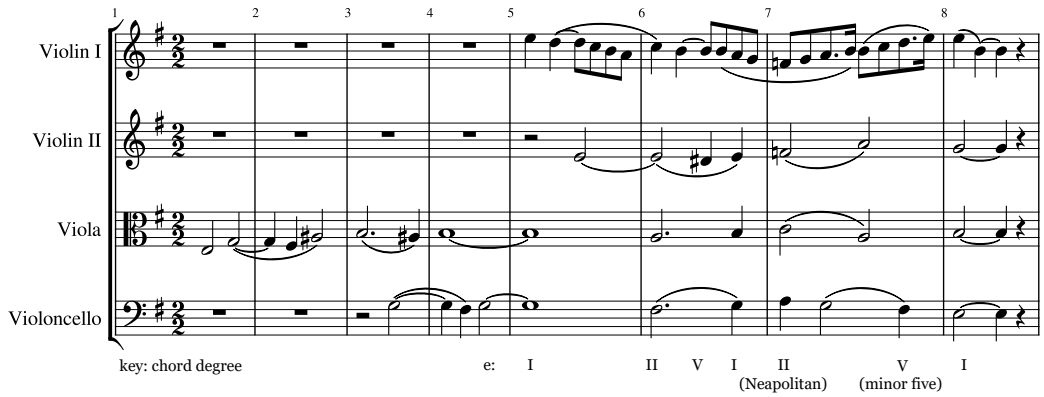
When the chorale melody begins (on and after the last beat of bar 5), listeners hear an artistic contrast; the key is settled on **D** major, and the chord progressions follow the principle of chord functions, accompanied by decorative secondary dominants. This satisfactory chord progression (that ends with an authentic cadence) and rich texture powerfully express the bright meanings of the chorale text.

As seen above, the beginning part of the organ chorale was not described well with the typical chord progression rules. Conversely, it would be a polished technique by J. Brahms that produced the great contrast, reliant on trained ears with traditional chord progressions.

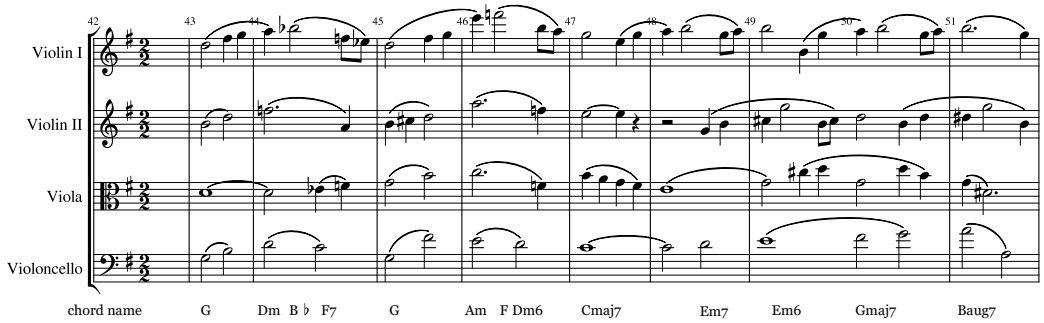
2.3 Questions to the Conventional Theory

A further example with G. Fauré's string quartet raises a fundamental question of what is chord, key, and function. The beginning part (**Figure 2.3a**) is still understandable with chord functions, though the transformed chords of Neapolitan **II** and minor five make it sound different from usual. On the other hand, it is hard to analyze bars 43–51 of the piece (**Figure 2.3b**) with the traditional harmony theory. The convention of representing a chord by its scale degree is based on the principle that chords are composed only with pitches on a particular key, arranged as tertian harmony. Although chords on different keys, especially secondary dominants, are commonly used as borrowed chords, others are difficult to be represented with the chord degree system.

As noted above, the notion of a chord and a chord function depends on a key. However, on the contrary, chord functions indicate a tonic of a particular key, in a situation where closely related keys are inherently ambiguous.



(a) Bar 1–8



(b) Bar 43–51

Figure 2.3: G. Fauré: String quartet Op.121

Therefore, H. Riemann discussed that a role (*i.e.,* chord function) change of a chord implied a modulation [70]. For example, a chord progression of $\mathsf{G} \rightarrow \mathsf{C}$ would be highly ambiguous between $\mathbf{V} \rightarrow \mathbf{I}$ in **C** major key and $\mathbf{I} \rightarrow \mathbf{IV}$ in **G** major key [77] [1]. In this case, **C** major key is more likely since the progression of the dominant (D) → tonic (T) is assumed to be the strongest among the functional chord progressions. This "chicken and egg" relationship between keys and chord functions makes such a piece challenging to be analyzed that is not based on a distinct key or stereotypical chord progressions.

Figure 2.3b would not be adequately analyzed in a conventional theory, although some characteristics of the piece could be interpreted with known concepts such as a seamless interchange between relative keys and borrowed chords from the parallel minor. However, this piece also has some sense of modal harmony, making it challenging to align to a particular major or minor key. At present, we have given up assigning chord degrees to it and just shown a possible interpretation of chord names. In this case, even the assignment

[1] V7 (dominant seventh) does not have this problem since it has F [77].

of chord names is not trivial because of finely meshed suspensions.

As we have seen thus far, masterpieces extend far beyond a textbook harmony theory. Although it would be impossible to know all of their sophisticated techniques, the motivation behind this study is to find the characteristics of their art without fitting them into an incomplete stereotype.

Chapter 3

Background

3.1 Statistical Learning for Music

Statistical learning of music has traditionally been developed for individual concepts such as key detection and chord function identification. While this study aims for a unified recognition of harmony, function, and tonality, it also draws on previous studies of individual elements. This section reviews such related studies for key detection and chord function identification.

3.1.1 Key or mode detection

Most key-finding algorithms were based on the frequency of each pitch-class represented as a histogram [1, 2, 37, 42, 51, 79]. An optimal key is detected by comparing the histogram with a template that is called key-profile. One of the most well-known works is the Krumhansl-Schmuckler algorithm which adopts the key-profile obtained by a psychological experiment [51]. More recent works employed statistical, data-oriented approaches. Temperley constructed it by a simple count-based probabilistic model and showed that this data-oriented model could estimate some tonal properties [77][1]. Albrecht and Shanahan used large corpora and improved the detection accuracy [2]. Hu and Saul employed a topic model based on Latent Dirichlet Allocation (LDA) and trained key profiles by unsupervised learning [42]. Assuming that the number of keys was 24 and relative homogeneity between keys sharing the same scale, the model automatically obtained key profiles corresponding to major and minor scales.

However, pitch-histogram-based results would be ambiguous when there is no difference in the observed count of pitch-classes. For example, a chord

[1] They represented them as *tonal implication*, *tonal ambiguity*, and *tonalness*.

progression of $\mathsf{G} \rightarrow \mathsf{C}$ would be highly ambiguous between $\mathbf{V} \rightarrow \mathbf{I}$ in $\mathbf{C}$ major key and $\mathbf{I} \rightarrow \mathbf{IV}$ in $\mathbf{G}$ major key [78]. Therefore, several works have considered chord progressions for key detection [26, 56]. The advantage of using chord progressions is that a strong functional progression such as the dominant motion would strongly suggest a particular key without being restricted by fixed scope where a pitch-class histogram is calculated. Thus, it was found to be effective for modulation detection [56, 84].

Most preceding works of key-finding algorithms have presupposed the modern tonal system of 24 keys [2, 6, 42, 51, 78, 79]. Therefore, several data-oriented computational music analysis works have been excluded or manually normalized pieces written in church mode [73, 84, 88]. However, excluding pieces that do not follow the modern tonal system would conflict with the concept of data-oriented music analysis aiming to reflect the diversity of targeted data rather than relying on textbook knowledge. Only a few computational studies have focused on finding modes to the best of our knowledge. Harasim et al. built a histogram-based Bayesian probabilistic model and adapted it to datasets that were divided by time periods [37]. As a result, they found four clusters of modes for the Renaissance era and two for later eras.

3.1.2 Chord function identification

Several studies have endeavored to find the statistical property of harmony or chord progressions in a data-driven manner rather than relying on textbook knowledge [44, 73, 82, 88].

Rohrmeier and Cross adopted a hierarchical clustering on bi-grams of pitch-class sets from J. S. Bach's four-part chorales, where major and minor pieces were divided. They found that chords that preceded dominant were the most distinctive in both major and minor scale datasets. In addition, asymmetric properties between major and minor keys, such as relatively weak tonic functionality in minor, also emerged [73]. This result is understandable since the minor scale retains the characteristics of medieval church modes [75]. Although their study was illuminating [73], using a heuristic chord segmentation as a pre-process would affect the clustering result and become an obstacle to the broader application of the data-oriented approach.

Jacoby et al., therefore, investigated the balance of the chord clustering by introducing the optimal complexity-accuracy curve [44]. They conducted the chord clustering with several traditions of chord representations, such as chord degrees and figured-bass representations. When using chord degrees as chord classes, the clustering with functional harmony theory based on tonic, dominant, and subdominant was plotted on the optimal curve. In contrast, clustering with the chord qualities (*i.e.,* major, minor, and diminished) was

far less accurate. These results suggest that the functional harmony theory is more favorable than chord qualities as categorization for chords when using the chord degrees as the chord representations.

On the other hand, Tsushima et al. [82] found the chord functions from Berklee chord sequences of a popular music dataset by modeling chord progressions with generative models (hidden Markov model (HMM) and probabilistic context-free grammar (PCFG)) rather than clustering. They reported that when the number of states was 4, the trained output probability could be interpreted as the chord functions: tonic, dominant, subdominant, and others, though the model achieved better *perplexity* with more states.

White and Quinn also employed HMM in datasets that include J. S. Bach's four-part chorales and found that obtained clusters and their state transition property showed similarities to the known chord functions [88]. When employing HMM, selecting an optimal number of clusters is not trivial; a larger number of clusters would help the model gains accuracy but loses simplicity and interpretability. However, selecting an optimal number of clusters would not be trivial since the larger number of clusters would help the model to gain accuracy but lose simplicity or interpretability. Therefore, they adapted the *k-medoids* clustering to find the best number of states for HMMs; the optimal numbers of hidden states were 3 and 13.

As we have seen, the previous studies above successfully found the chord functions consistent with general harmony theory. However, they performed chord clustering under limited conditions; major/minor pieces are separated [73, 44], or only majors are used after modulation segmentation [88]. In other words, they avoided the inherent ambiguity of chords and tonalities by pre-processing. Such pre-processing would have helped the model obtain coincided results with known chord functions. Conversely, Hugo Riemann discussed that a role change of a chord in a context implied a modulation [70]; this suggests that the independent treatments of chords and tonalities would not follow the nature of music. Furthermore, more complicated pieces would not be trivially pre-processed. Therefore, this study seeks a more unified recognition of chord, function, and tonalities, which eliminates heuristic pre-processing as much as possible.

Finally, we describe some background as to why we use a relatively simple model, a kind of (semi-)Markov model, in this study. In applications such as the generation of chord progressions [62] or melody harmonization [81], tree-structured models were shown to be more advantageous. Nevertheless, as the most closely related works suggest [73, 44, 82, 88], bi-grams or Markov property of chord progressions well agrees with the functional harmony theory. We probe into the functional property of chord progressions without blurring the issue rather than employ a model with high expressive power.

We agree that HMM cannot discuss musical forms that have a structure of remote dependency. At the same time, we expect that these remote dependencies would not be described based solely on chord progressions but also on metrical regularities and melodic structures. Although these remote dependencies will not be addressed in this study, we will incorporate the chord length, which has been ignored in previous studies, utilizing a semi-Markov model [90] because we consider it important in modeling chord progressions.

3.2 J. S. Bach's Four-part Chorales

In this dissertation, we consult J. S. Bach's four-part choral pieces BWV253–438 from the Music21 corpus, formatted as MusicXML [21][2]. These chorales are indexed by the Riemenschneider numbering system [71], the total number of which is 371.

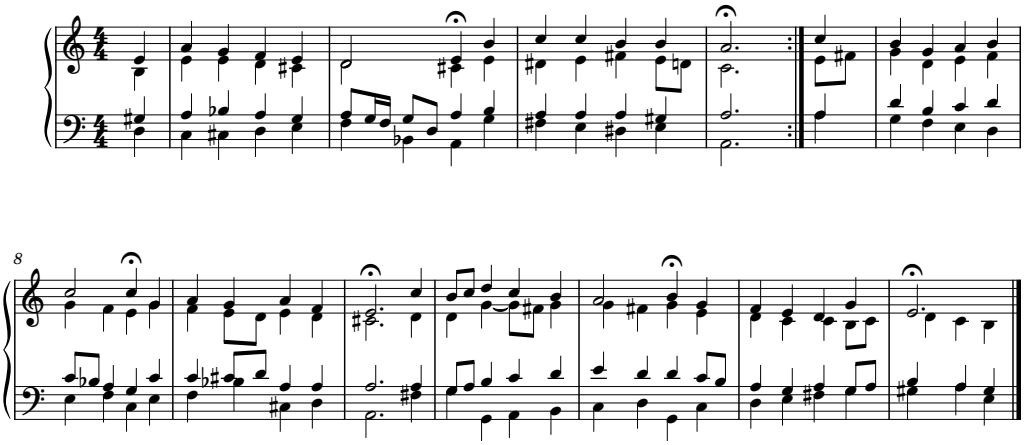
Chorales are originally hymns sung in Lutheran Church, the texts of which are written in German. Most hymn melodies in the dataset were not composed by J. S. Bach. The melodies originated from Gregorian chants, folk songs, and other familiar melodies [71]. Even Martin Luther (1483–1546) himself composed texts and melodies for some of them. The four-part chorales is a collection of harmonizations by J. S. Bach based on such chorale melodies. Thus, the melodies retain the feature of the medieval to Renaissance era. Therefore, some are not written in modern 24 keys, but in medieval church modes, especially *dorian* [22, 73].

The four-part chorales were not performed in *a cappella* but accompanied by an orchestra in real, larger-scale pieces. Nevertheless, the outstanding art by J. S. Bach can be found in the harmonization, especially in the bass lines. In addition, he often used unusual harmonic progressions to express the meanings of the chorale texts [71].

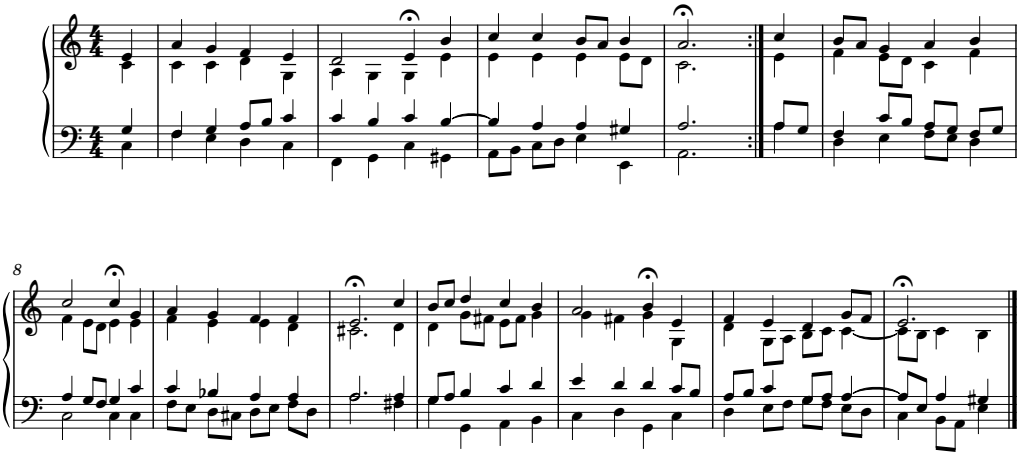
Thus, the harmonized four-part chorales, even though in the simplest form, cannot be fully analyzed by the generalized harmony theory. Therefore, we believe this collection is an appropriate starting point for our data-driven harmonic analysis.

According to the analysis by Dahn [22], there are several duplications in the 371 harmonized chorales; thus, we have removed those 23 pieces based on it. However, we regard different harmonizations as independent items, examples of which are shown in **Figure 3.1**. Although these pieces share the same melody, their harmonizations are different. Interestingly, even the key for the beginning section seems to be different.

[2]In works for melody harmonization, the JSB Chorale dataset [13] that only includes MIDI numbers is often used.



(a) Riemenschneider No.21



(b) Riemenschneider No.367. The key is normalized so as not to have the key signature.

Figure 3.1: Examples of different harmonizations on the same melody.

The *fermata* notation (point d'orgue) represents the end of a lyric paragraph in chorales. We regard each phrase divided at the *fermata* position as an independent sequence; however, when we provide these phrases to the training, evaluation, and testing set, we give randomness over pieces (not over phrases).

3.3 Technically Related Works

This section introduces technically related models and works. Firstly, we describe hidden Markov model (HMM) in the next section (Section 3.3.1), then hidden semi-Markov model (HSMM) (Section 3.3.2), which is the core

model of this study. Next, we introduce perplexity that is the automatic evaluation metric for statistical models in Section 3.3.3. We utilize neural networks to embed additional contexts to the HSMM; related neural network components are described in Section 3.3.4. Finally, we review related works of unsupervised part-of-speech (POS) induction, from which we have drawn technical ideas.

3.3.1 Hidden Markov Model

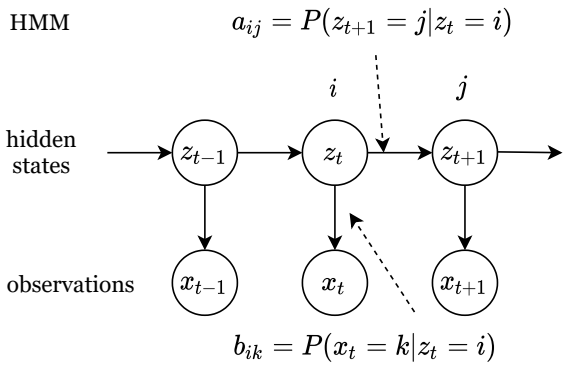


Figure 3.2: Graphical representation of hidden Markov model (HMM).

hidden Markov model (HMM) is a simple but beneficial model employed in numerous applications such as speech recognition [29], biotechnology [25], economics [38], and music [64, 83]. In the model, a Markov chain of hidden states lies behind an observable sequence. Therefore, it has been used especially in sequence labeling tasks where hidden states represent labels to be assigned. HMM can be trained through both supervised and unsupervised learning [68]. Supervised learning can be adopted when labeled data for hidden state sequences are available. On the other hand, unsupervised learning that optimizes the log marginal likelihood is usable when we do not have any manually labeled data.

The joint probability of the HMM is expressed as follows,

$$P(\mathbf{x}_{1:T}, \mathbf{z}_{0:T}) = P(z_0) \prod_{t=1}^{T} P(z_t | z_{t-1}) P(x_t | z_t)$$

where z_t is a hidden state and x_t is an observation at time t. Thus, HMM consists of the following three categorical distributions[3]:

[3]Emission distribution is not limited to the categorical distribution.

Initial hidden state probability: $\rho_i = P(z_0 = i)$

Hidden state transition probability: $a_{ij} = P(z_t = j|z_{t-1} = i)$

Emission probability: $b_{ik} = P(x_t = k|z_t = i)$

where i, j are hidden state indices, and k is the index for an observed symbol. Once the model is trained, the best hidden state sequence is decoded by the Viterbi algorithm [28].

The marginal probability (3.1) for HMM is calculated through a dynamic program called the *forward-backward* algorithm.

$$P(\mathbf{x}_{1:T}) = \sum_{\mathbf{z}_{0:T}} P(\mathbf{x}_{1:T}, \mathbf{z}_{0:T}) \tag{3.1}$$

In order to describe the algorithm, the forward probability $\alpha_t(j)$ and the backward probability $\beta_t(j)$ are introduced as follows [10, 68].

$$P(z_t = j|\mathbf{x}_{1:T}) = \frac{P(z_t = j, x_1, ..., x_t)P(x_{t+1}, ..., x_T|z_t = j)}{P(\mathbf{x}_{1:T})} = \frac{\alpha_t(j)\beta_t(j)}{P(\mathbf{x}_{1:T})}$$
$$\alpha_t(j) \equiv P(z_t = j, x_1, ..., x_t)$$
$$\beta_t(j) \equiv P(x_{t+1}, ..., x_T|z_t = j)$$

Both $\alpha(z_t)$ and $\beta(z_t)$ can be calculated recursively as shown in (3.2) and (3.3). The calculating processes are called *forward* algorithm and *backward* algorithm respectively.

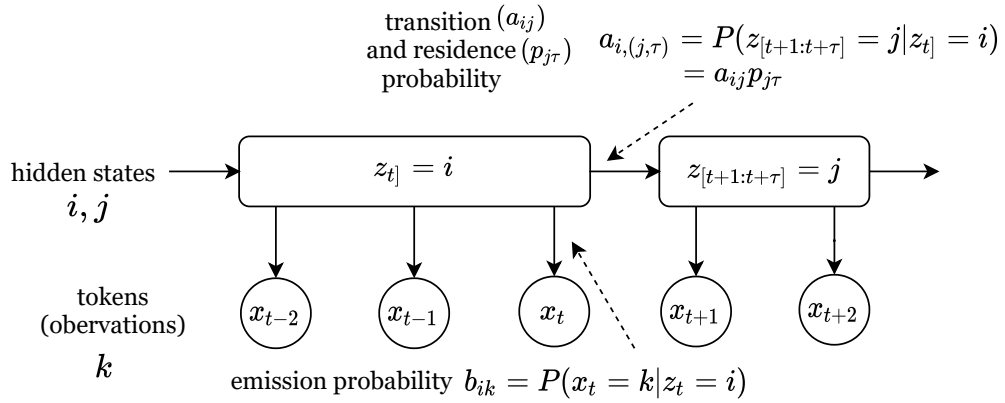
$$\text{Forward: } \alpha_t(j) = P(x_t = k|z_t = j) \sum_i \alpha_{t-1}(i)P(z_t = j|z_{t-1} = i) \tag{3.2}$$

$$\text{Backward: } \beta_t(j) = \sum_i \beta_{t+1}(i)P(x_{t+1} = k|z_{t+1} = i)P(z_{t+1} = i|z_t = j) \tag{3.3}$$

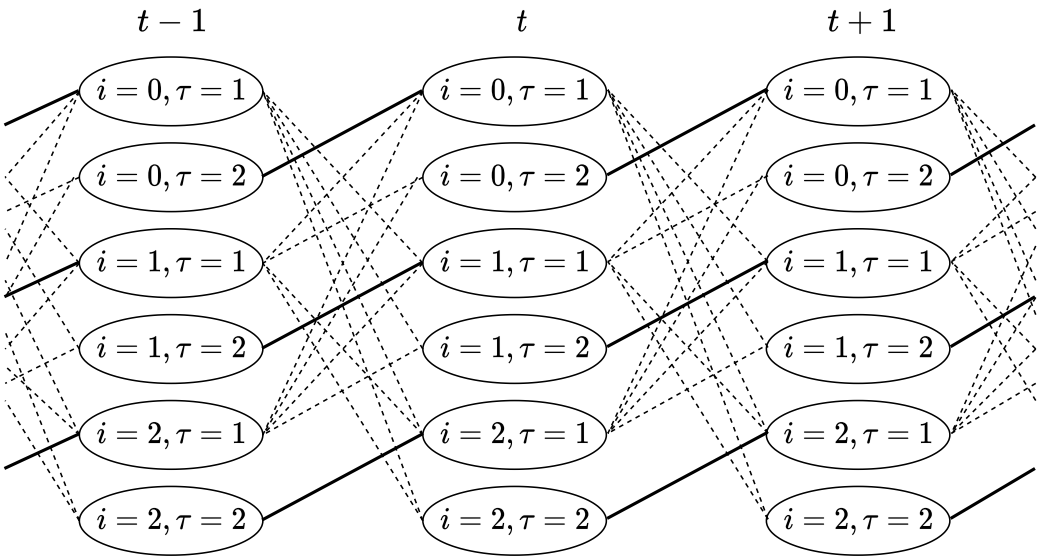
Since $\sum_i P(z_t = i|\mathbf{x}_{1:T}) = 1$, the marginal probability $P(\mathbf{x}_{1:T})$ is calculated as follows: where t can be selected arbitrarily. If we only need to know the marginal probability, we can select T, which eliminates the need to calculate the backward recursion since $\beta_T(i) = 1$ [10].

$$P(\mathbf{x}_{1:T}) = \sum_i \alpha_t(i)\beta_t(i) = \sum_i \alpha_T(i)$$

Although both *forward* and *backward* recursions are employed in the expectation maximization (EM) algorithm for unsupervised training of HMM [10, 68], we directly optimize the marginal probability by a gradient-based optimizer and thus only require the forward algorithm.



(a) Graphical representation of Hidden Semi-Markov Model (Residential-time HMM). The notation $z_{:t]}$ means a hidden state z ends at time t, similarly, $z_{[t+1:t+\tau]}$ starts at time $t+1$ and ends at time $t+\tau$.



(b) Possible paths of hidden states and durations for HSMM (Residential-time HMM). In this example, the number of hidden states is 3, and the maximum duration of a hidden state is 2. The dashed line represents the state index change and the duration selection. The solid line represents the decrement of the state duration.

Figure 3.3: Hidden Semi-Markov model.

3.3.2 Hidden Semi-Markov Model

Hidden semi-Markov model (HSMM) extends hidden Markov model (HMM) by introducing the notion of the duration of each hidden state. In other words, while HMM implicitly represents the state duration by the increase of the ratio of self-transition, HSMM equips an explicit architecture for it. This is important for us when we do not have chord labeling or segmentation. Although there are several variants for HSMM of the difference in modeling about the duration [90], we select a model called "Residential-time HMM" [91].

Graphical representation of HSMM (Residential-time HMM) is shown in **Figure 3.3a**, where we borrow the notation used by Yu et al. [90].

$t_1 : t_2]$: a state lasts at latest from t_1 and ends at time t_2.

$[t_1 : t_2]$: a state starts at t_1 and ends at t_2 (with duration $t_2 - t_1 + 1$).

Residential-time HMM assumes that a hidden state transition is independent of the duration of the previous hidden state. With this assumption, hidden state transition is described as follows[4].

$$P(S_t = (j, \tau')|S_{t-1} = (i, \tau)) = \begin{cases} a_{i,(j,\tau')} & \text{if } \tau = 1 \text{ (transition)} \\ \mathbb{1}(\tau' = \tau - 1) & \text{if } \tau > 1 \text{ (decrement)} \end{cases}$$

Therefore, possible paths consist of products of hidden states and durations as shown in **Figure 3.3b**, where i is an index of hidden state and τ is the remaining duration of it. Furthermore, $a_{i,(j,\tau')}$ can be decomposed into transition probability and duration probability as follows.

$$a_{i,(j,\tau')} = a_{ij} p_{j\tau'}$$

Thus, in HSMM, duration probability[5] is added in addition to the three categorical distributions for HMM described in Section 3.3.1. Note that a hidden state changes to another only when the remaining duration $\tau = 1$. Therefore, the duration probability determines a hidden state duration, and the self-transition probability a_{ii} is always zero.

Forward Algorithm for HSMM

We have seen that the marginal probability $\ln P(\mathbf{x}_{1:T})$ for HMM is calculated through the forward algorithm. In this paragraph, we detail the one for HSMM.

[4] $\mathbb{1}(x)$ is the indicator function that equals to 1 if x is *true* and 0 otherwise.

[5] $\tau \in D$ is a discrete value of remaining time steps, and D is the maximum duration.

For HSMM, the forward algorithm marginalizes the possible paths of durations in addition to states (as illustrated in **Figure 3.3b**). The $\alpha_t(j,\tau)$ represents the forward probability that the hidden state at t is j and the remaining duration of which is τ, and output tokens from $t=1$ to t, *i.e.*, $\mathbf{x}_{1:t}$, are observed. It is decomposed as follows:

$$\begin{aligned}
\alpha_t(j,\tau) &= P(z_{t:t+\tau-1]} = j, \mathbf{x}_{1:t}) \\
&= \alpha_{t-1}(j,\tau+1)P(x_t = k|z_t = j) \\
&\quad + P(\tau|z_t = j)P(x_t = k|z_t = j)\sum_{i\setminus j}\alpha_{t-1}(i,1)P(z_t = j|z_{t-1} = i) \\
&= \alpha_{t-1}(j,\tau+1)b_{jk} + p_{j\tau}b_{jk}\sum_{i\setminus j}\alpha_{t-1}(i,1)a_{ij}
\end{aligned}$$

where $P(\tau|z_t = j) = p_{j\tau}$ is duration probability, $P(x_t = k|z_t = j) = b_{jk}$ is emission probability, and $P(z_t = j|z_{t-1} = i) = a_{ij}$ is transition probability. Note that when the state at t is j, there are two possibilities: (i) the hidden state at time $t-1$ is also j with the remaining duration at there is $\tau+1$, (ii) the hidden state at time $t-1$ is i ($i \neq j$) and the remaining duration is 1 then is transferred another hidden state j at the time t.

We apply a *scaling* by replacing $\alpha_t(i,\tau)$ to the conditional probability, similar to HMM [90], and obtain a modified forward algorithm.

$$\begin{aligned}
\hat{\alpha}_t(j,\tau) &= P(z_{t:t+\tau-1]} = j|\mathbf{x}_{1:t}) = \frac{\alpha_t(j,\tau)}{P(x_1,\ldots,x_t)} \\
C_t &= P(x_t|x_1,\ldots,x_{t-1}) \\
P(x_1,\ldots,x_t) &= \prod_{t'}^{t} C_{t'} \qquad (3.4) \\
C_t\hat{\alpha}_t(j,\tau) &= \frac{\alpha_t(j,\tau)}{P(x_1,\ldots x_{t-1})} \\
&= \hat{\alpha}_{t-1}(j,\tau+1)b_{jk} + p_{i\tau}b_{jk}\sum_{i\setminus j}\hat{\alpha}_{t-1}(i,1)a_{ij} \qquad (3.5)
\end{aligned}$$

As can be seen from (3.4), the marginal probability is obtained by $P(\mathbf{x}_{1:T}) = P(x_1,\ldots,x_T) = \prod_{t'}^{T} C_{t'}$, where T is the length of an entire observation sequence. Note that C_t is obtained by summing up (3.5) about all j and τ, since $\sum_j\sum_\tau \hat{\alpha}_t(j,\tau) = 1$. We set the initial probability to yield the initial hidden state z_0 that does not yield observation by considering the initial boundary condition: $\alpha_0(i,1) = P(z_0 = i)$ and $\alpha_0(i,\tau) = 0$ for $\tau > 1$ [90]. Therefore, $\alpha_0 = \hat{\alpha}_0$ and $C_0 = 1.0$ at $t = 0$.

3.3.3 Perplexity

Unsupervised learning is a technique to discover hidden patterns from raw data without human supervision. Manually annotated test data are not always available. Furthermore, obtaining consistent chord annotations or key assignments is not trivial in the music domain [50, 77].

In order to evaluate unsupervised sequential modeling without any reference data, *perplexity* is used as a common metric [5, 15, 16, 47, 82]. Perplexity can be expressed in several equivalent forms:

$$\text{Perplexity: } \mathcal{P} = \sqrt[T]{\frac{1}{P(\mathbf{x}_{1:T})}} \tag{3.6}$$

$$= 2^{-\frac{1}{T}\log_2 P(\mathbf{x}_{1:T};\boldsymbol{\theta})} \tag{3.7}$$

$$= \exp\left(-\frac{1}{T}\ln P(\mathbf{x}_{1:T};\boldsymbol{\theta})\right)$$

where $\mathbf{x}_{1:T}$ is an observed sequence, and $\boldsymbol{\theta}$ is a set of model parameters. As can be seen in (3.6), perplexity is the inverse probability normalized by the sequence length. It is generally desirable that a trained model achieves high test data probability, and thus smaller perplexity means better generalization performance.

In addition, perplexity has a relation to the *cross-entropy* [47]. When the actual probability is P^* (that cannot be known), cross-entropy between it and an estimated probability is as follows.

$$H(P^*, P) = \lim_{T\to\infty} -\frac{1}{T}\sum_{\mathbf{x}_{1:T}} P^*(\mathbf{x}_{1:T})\log_2 P(\mathbf{x}_{1:T}) \tag{3.8}$$

However, according to the Shannon-McMillan-Breiman theorem, (3.8) is approximated into (3.9) if the process is stationary and ergodic [20, 47].

$$H(P^*, P) = \lim_{T\to\infty} -\frac{1}{T}\log_2 P(\mathbf{x}_{1:T}) \tag{3.9}$$

Thus, we can obtain perplexity by raising 2 to the power of the cross-entropy, as can be seen from (3.7) and (3.9).

3.3.4 Neural Networks

This section briefly introduces the neural network components employed in this work.

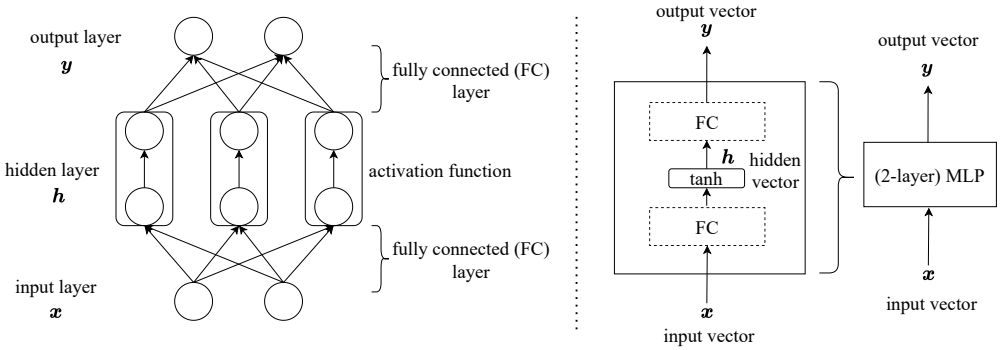


Figure 3.4: Multi Layer Perceptron (MLP) or Feedforward Neural Network (FNN).

Multi Layer Perceptron

Multi layer perceptron (MLP) or feedforward neural network (FNN) is the essential neural network, which works as a function $\boldsymbol{y} = f(\boldsymbol{x}, \theta)$ [33]. An MLP consists of an input layer, (a) hidden layer(s), and an output layer. However, from the perspective of learnable components, an MLP with one hidden layer equips two fully connected layers, and thus it is called "2-layer" MLP. Similarly, an MLP with two hidden layers is called "3-layer" MLP, and so on.

A fully connected layer applies a linear transformation[6] as follows,

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

where $\boldsymbol{W}$ is a learnable weight matrix and $\boldsymbol{b}$ a bias vector.

An activation function that is a nonlinear function usually follows each hidden layer, which plays an important role for MLP to be a good approximate function. In this dissertation, we use the hyperbolic tangent function (tanh) as the activation function for MLPs, which normalizes the value to the range -1 to 1.

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Thus, the whole process of 2-layer MLP is as follows.

$$\mathrm{MLP}_2(\boldsymbol{x}) = \boldsymbol{W}^{(2)}(\tanh(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})) + \boldsymbol{b}^{(2)}$$

When we need a categorical distribution as output, we further apply the

[6]It is called an affine transformation since it has a bias term.

softmax function to the output layer,

$$\text{softmax}_j(\boldsymbol{x}) = \frac{\exp(x_j)}{\sum_{j'}^{K} \exp(x_{j'})} \tag{3.10}$$

where K is the number of categories.

Recurrent Neural Network

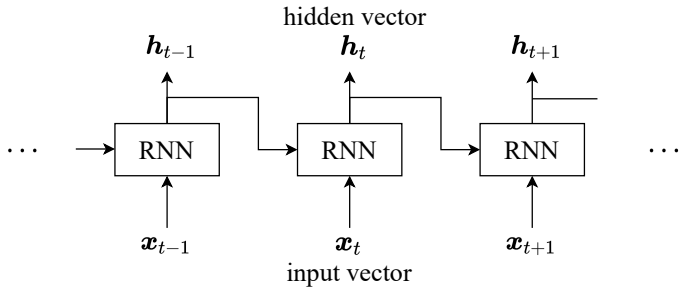


Figure 3.5: Recurrent Neural Network (RNN).

Recurrent Neural Network (RNN) recursively embeds produced hidden vectors as inputs, as shown in **Figure 3.6**, and thus it is suitable for sequential data. Each RNN unit is similar to a fully connected layer with the tanh activation function.

$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}^{in}\boldsymbol{x}_t + \boldsymbol{b}^{in} + \boldsymbol{W}^{rec}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{rec})$$

Long-Short Term Memory

In practice, the vanilla RNN described in the previous paragraph is rarely used since it is known to suffer the exploding and vanishing gradient problem [65]. Long-Short Term Memory (LSTM) [41] improves the architecture by introducing gates to control the flow of information.

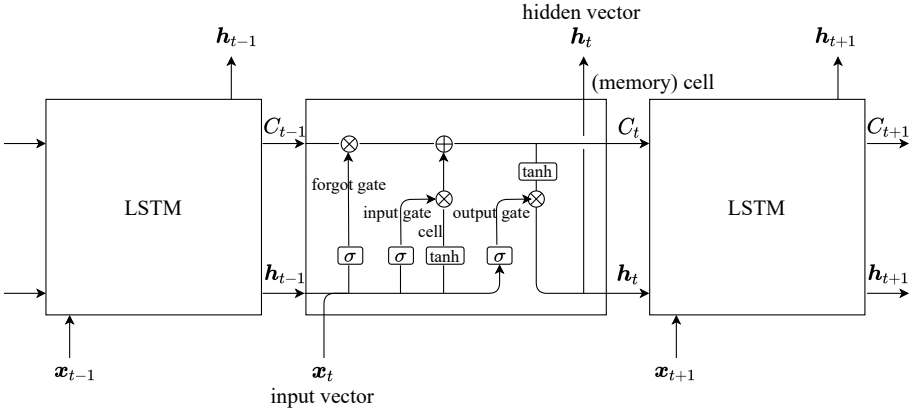


Figure 3.6: Long-Short Term Memory (LSTM).

For each time step, the LSTM unit calculates the following functions[7]:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{f_x}\boldsymbol{x}_t + \boldsymbol{b}^{f_x} + \boldsymbol{W}^{f_h}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{f_h}) : \text{forget gate} \tag{3.11}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{i_x}\boldsymbol{x}_t + \boldsymbol{b}^{i_x} + \boldsymbol{W}^{i_h}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{i_h}) : \text{input gate} \tag{3.12}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{o_x}\boldsymbol{x}_t + \boldsymbol{b}^{o_x} + \boldsymbol{W}^{o_h}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{o_h}) : \text{output gate} \tag{3.13}$$

$$\boldsymbol{g}_t = \tanh(\boldsymbol{W}^{g_x}\boldsymbol{x}_t + \boldsymbol{b}^{g_x} + \boldsymbol{W}^{g_h}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{g_h}) : \text{cell} \tag{3.14}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{3.15}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{3.16}$$

where σ is the *sigmoid* activation function[8], and $\odot$ denotes Hadamard product. The three gates, *i.e.,* forget ($\boldsymbol{f}_t$), input ($\boldsymbol{i}_t$), and output ($\boldsymbol{o}_t$), work as gates to control the amount of information to be added or discarded, and thus the sigmoid function that normalizes the value 0 to 1 is applied. On the other hand, $\boldsymbol{g}_t$ is an information embedding, and thus tanh is employed as the activation function just like the vanilla RNN.

We use $\boldsymbol{h}_t$ as an embedded feature for a sequence observed up to t. Since $\boldsymbol{c}_t$ is used only internally, we simplify the notation for LSTM as follows to mean all processes of (3.11),(3.12),(3.13),(3.14),(3.15), and (3.16).

$$\boldsymbol{h}_t = \text{LSTM}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

[7]Although there are several variants for LSTM, we employ the PyTorch implementation [66].

[8]$\sigma(x) = \frac{1}{1+\exp(-x)}$

3.3.5 Unsupervised Part-of-Speech Induction

Unsupervised chord function identification is technically closely related to unsupervised part-of-speech (POS) induction in Natural Language Processing (NLP); the task is to identify latent categories of words that are expected to be part-of-speech.

Brown et al. [14] proposed a greedy clustering method to maximize the average mutual information of adjacent word classes (bigram of classes) and obtained clusters that were adequate both syntactically and semantically. They also constructed an interpolated class-based 3-gram language model by using the assignment of obtained clusters. The class-based model achieved better perplexity than the word-based 3-gram model, though the improvement was small. Nevertheless, this pioneering work showed that a simple feature, the count of bi-grams, could provide syntactic and semantic aspects of natural language.

HMM is a natural selection for building language models with latent word classes. The expectation maximization (EM) algorithm is widely used to train HMM but is known to be often stuck into local optima. Johnson [45] employed hidden Markov models (HMMs) for building the language model and tested multiple optimization methods: Expectation maximization (EM), Gibbs sampling, and Variational Bayes. The results showed that EM assigned a relatively equal number of words to each hidden state and scored worse than other methods in terms of 1-to-1 accuracy [9]. Similarly, Goldwater and Griffiths [32] showed the efficacy of Bayesian methods for POS tag induction, which make use of priors that are likely to produce sparse distributions. In addition, the full Bayesian methods integrate over possible parameters and thus are more robust than Maximum likelihood estimation (MLE) trained by EM.

In order to improve models for POS tag induction, the effectiveness of utilizing linguistic features such as morphological information has been shown. Clark [19] significantly improved the perplexity of the class-based language model by incorporating morphological information. Berg-Kirkpatrick et al. [8] proposed an extension of HMMs that incorporated manually designed additional features by modeling emission distribution as softmax regression [10]. Although a modified EM algorithm could be applied for training the model, directly optimizing the likelihood was empirically more effective than the EM.

More recently, Tran et al. proposed the unsupervised neural HMM [80];

[9] 1-to-1 accuracy is an evaluation metric, where at most one cluster can be assigned to any gold standard tag.

[10] Softmax regression is a multi-class extension of logistic regression.

the model retained the strength of seamless integration with additional contexts. They introduced two additional contexts: infinite context with the Long-Short Term Memory (LSTM) [11] and morphological information via character Convolutional Neural Networks (CNNs). Unlike the feature HMM [8] that incorporated manually designed features, the neural HMM automatically utilized additional contexts to calculate transition and emission distributions with the help of neural networks such as LSTM and CNN. Despite its simple framework, the neural HMM outperformed even the highly polished models such as the Bayesian mixture model [18] or the hierarchical Pitman-Yor Process HMM [12].

Note that the neural HMM can be regarded as a variant of Input-Output HMM [7] if we consider the additional contexts as input features. In addition, unlike the Hybrid DNN-HMM[12] model [89] that converts a pre-trained GMM-HMM[13] to a DNN-HMM or the tandem model [39] that combines features by a supervised DNN to an HMM, the neural HMM is seamless and fully unsupervised.

[11] They called it "infinite" since the LSTM could embed an unlimited length of preceding observations.

[12] Deep Neural Network (DNN)

[13] Gaussian Mixture Model (GMM)

Chapter 4

Automatic Chord Segmentation and Chord Function Recognition by Neural HSMM

In this chapter and the following one, we present our methodology of unsupervised recognition of chords, functions, and tonality; these three are central notions of harmony theory. As previously stated in Section 1.2, we regard these notions as corresponding to the following statistical learning.

1. Chords are recognized by automatic segmentation and categorization of surface pitch events.

2. Chord functions are statistical properties of transitions between chord categories.

3. Tonality or key recognition is a categorization of chord transitions.

In this chapter, we introduce a model that automatically learns chord categories and segments surface notes to be classified into them; in other words, a model that performs 1. and 2. Instead of relying on pre-defined chord symbols and keys, this study employs unsupervised learning to categorize chords; we expect an appropriate set of chords to differ by musical style and aim to obtain them in a data-driven manner.

Following the previous studies [82, 88], we select an hidden Markov model (HMM) as a promising one to analyze chord progressions. We take over a popular assumption of transpositional equivalency between keys, which is employed in general music theories and previous studies [42, 44, 51, 73, 75, 88]. Therefore, the targeted scores are transposed not to have key signatures before the analysis. Even though the main keys are normalized, a wide

variety of local modulations are expected to exist. As H. Riemann pointed out [70], chord functions are changed by local modulations; a single transition matrix of the conventional HMM employed in previous studies would be insufficient to analyze chord progressions. Therefore, we employ the idea of neural HMM, which can adjust the hidden state transition probability by the contexts [80]. We can easily embed additional contexts with neural networks to improve transition and emission architectures. Furthermore, to make the model consider metrical structures, we extend the neural HMM to the neural hidden semi-Markov model (HSMM).

We evaluate our model on J. S. Bach's four-part chorales dataset (→ Section 3.2) and use *perplexity* (→ Section 3.3.3) as the evaluation metric based on previous works [82, 84]. Experiments show that our model appropriately segments and classifies surface pitch classes, especially with the smallest-perplexity model. Additional contexts with neural network modeling are shown to improve perplexity. In addition, we show that the transitions between categories reflect the difference in tonalities when we count them by separating pieces into groups of majors, minors, and *dorian* scales.

4.1 Framework

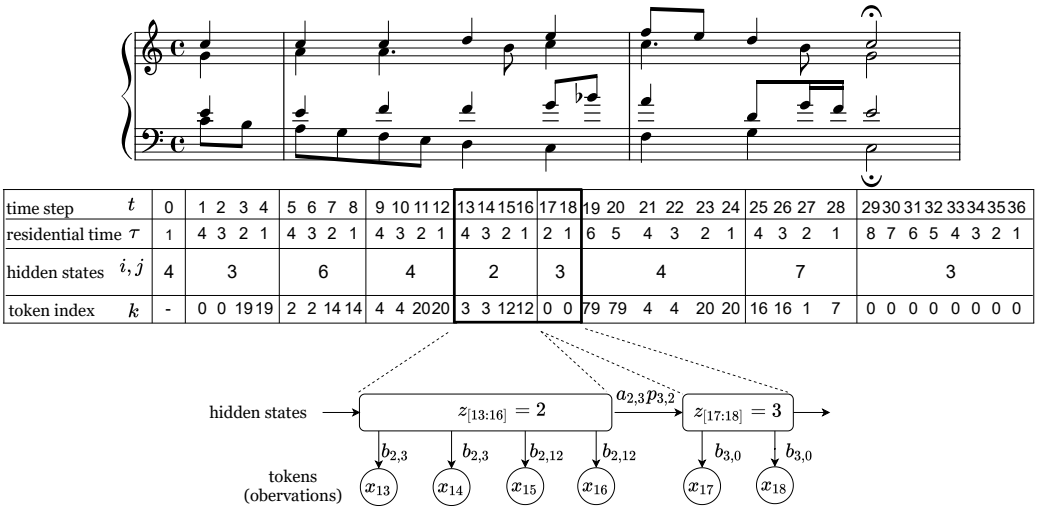


Figure 4.1: BWV294 (in evaluation set) of analysis by the proposed model (8-state HSMM). The key is transposed to have no key signature.

As seen in Section 3.3.2, hidden semi-Markov model (HSMM) is an extension of hidden Markov model (HMM), where a Markov chain of hidden

states lies behind an observable sequence. With the "semi-Markov" extension, we can consider the duration of each hidden state to avoid the useless increase of the ratio of self-transition. Although there are several variants for HSMM as to the difference in modeling about the duration, we select a model called "Residential-time HMM", which does not allow self-transition (→ Section 3.3.2).

In HSMM, each hidden state ($z_{t]} = i$) can produce multiple observations. Then, when a hidden state changes to another one (j), the model calculates the next state's duration probability $p_{j\tau}$ and the transition probability a_{ij}. However, there could be multiple possible combinations (or paths) of the hidden state and duration for HSMM, as we have seen in **Figure 3.3b**. For example, suppose the number of hidden states is three, and the maximum duration of a hidden state is two. When the hidden state index is $i = 0$, and the remaining duration is $\tau = 1$ at the time step t, possible conditions for the previous time step $t - 1$ are $(i = 0, \tau = 2)$, $(i = 1, \tau = 1)$, and $(i = 2, \tau = 1)$. Note that if the current remaining duration is $\tau = 1$, the hidden state must change to another at the next time step. On the other hand, if $\tau > 1$, the hidden state continues into the next time step.

We expect that hidden states represent chord categories, which are not given *a priori* and thus learned in an unsupervised manner. With the Markov property assumption, hidden states (chord categories) are learned to govern the progression of the forthcoming chord category. We argue that an appropriate set of chord symbols would be diverse along with targeted pieces and thus adopt unsupervised learning.

We give a more detailed description of the proposed framework by showing an example in **Figure 4.1**. Since HSMM takes discrete time series, we set a time step by every sixteenth note; it is the minimum duration of note in the corpus, except for only a few exceptions of 32nd notes. We obtain a pitch-class vector for each segment[1]. For example, when the time step is at 25, the pitch names of notes contained in the segment are (C, D, G), and thus corresponding pitch-class vector is $(0, 2, 7)$.

We build a vocabulary from every combination of the observed four-part pitch classes and call them tokens. For example, the token index for $(0, 2, 7)$ is set to 16. The whole vocabulary is shown in **Table 4.1**. As shown in the example, observations may include passing tones. We expect that a hidden state works as a chord category behind these raw pitch-class vectors. Looking at time step 25 again, we can see that the index of the hidden state is 7. Here, the hidden state lasts from time step 25 to 28, which includes three types of pitch-class vectors, $(0, 2, 7)$, $(2, 7, 11)$, and $(2, 5, 7, 11)$, but all the three can

[1] 12 pitch classes in an octave: C, C♯, D, ..., B are numbered as: 0, 1, 2, ..., and 11.

Table 4.1: Table of vocabulary.

index	pitch vec.	pitch name	index	pitch vec.	pitch name
0	(0,4,7)	C,E,G	40	(2,4,7,10)	D,E,G,A♯(Bb)
1	(2,7,11)	D,G,B	41	(1,4,7)	C♯(Db),E,G
2	(0,4,9)	C,E,A	42	(0,2,5)	C,D,F
3	(2,5,9)	D,F,A	43	(0,2,4,9)	C,D,E,A
4	(0,5,9)	C,F,A	44	(5,7,11)	F,G,B
5	(4,8,11)	E,G♯(Ab),B	45	(0,4,6,9)	C,E,F♯(Gb),A
6	(1,4,9)	C♯(Db),E,A	46	(0,5,7,9)	C,F,G,A
7	(2,5,7,11)	D,F,G,B	47	(1,4,7,10)	C♯(Db),E,G,A♯(Bb)
8	(2,6,9)	D,F♯(Gb),A	48	(0,7,9)	C,G,A
9	(4,7,11)	E,G,B	49	(2,4,7)	D,E,G
10	(0,2,5,9)	C,D,F,A	50	(4,7,10)	E,G,A♯(Bb)
11	(0,2,6,9)	C,D,F♯(Gb),A	51	(2,9,11)	D,A,B
12	(2,5,11)	D,F,B	52	(0,3,6,9)	C,D♯(Eb),F♯(Gb),A
13	(2,4,8,11)	D,E,G♯(Ab),B	53	(0,2,7,9)	C,D,G,A
14	(0,4,7,9)	C,E,G,A	54	(3,6,9,11)	D♯(Eb),F♯(Gb),A,B
15	(2,5,9,11)	D,F,A,B	55	(2,4,9,11)	D,E,A,B
16	(0,2,7)	C,D,G	56	(3,6,11)	D♯(Eb),F♯(Gb),B
17	()	Rest	57	(2,5,7,9)	D,F,G,A
18	(1,4,7,9)	C♯(Db),E,G,A	58	(2,7,9,11)	D,G,A,B
19	(0,4,7,11)	C,E,G,B	59	(2,6,11)	D,F♯(Gb),B
20	(0,4,5,9)	C,E,F,A	60	(2,5,7,10)	D,F,G,A♯(Bb)
21	(2,7,10)	D,G,A♯(Bb)	61	(0,2,9)	C,D,A
22	(0,2,4,7)	C,D,E,G	62	(0,2,6)	C,D,F♯(Gb)
23	(4,9,11)	E,A,B	63	(0,2,7,11)	C,D,G,B
24	(0,4,7,10)	C,E,G,A♯(Bb)	64	(2,4,7,9)	D,E,G,A
25	(2,4,7,11)	D,E,G,B	65	(4,5,9)	E,F,A
26	(0,6,9)	C,F♯(Gb),A	66	(2,5,7)	D,F,G
27	(2,8,11)	D,G♯(Ab),B	67	(0,7)	C,G
28	(2,4,9)	D,E,A	68	(2,4,8)	D,E,G♯(Ab)
29	(0,5,7)	C,F,G	69	(2,5)	D,F
30	(2,5,10)	D,F,A♯(Bb)	70	(0,4,11)	C,E,B
31	(2,4,5,9)	D,E,F,A	71	(0,5,9,11)	C,F,A,B
32	(2,5,8,11)	D,F,G♯(Ab),B	72	(5,9)	F,A
33	(0,2,5,7)	C,D,F,G	73	(4,7,9,11)	E,G,A,B
34	(0,4,9,11)	C,E,A,B	74	(2,5,9,10)	D,F,A,A♯(Bb)
35	(2,7,9)	D,G,A	75	(4,7,9)	E,G,A
36	(0,4)	C,E	76	(4,7)	E,G
37	(7,11)	G,B	77	(4,9)	E,A
38	(0,9)	C,A	78	(0,4,8)	C,E,G♯(Ab)
39	(0,4,5,7)	C,E,F,G	79	Others	Others

be interpreted as a part of G major chord.

A hidden state emits a token with probability b_{ik}, where i is a hidden state index and k is a token index. In the case of time step 25, pitch-class vector $(0, 2, 7)$ (the token index of which is 16) is emitted with the probability of $b_{7,16}$, from the hidden state indexed 7. A hidden state is changed to another one with the probability of transition probability a_{ij} multiplied by duration probability $p_{j\tau}$, where i is the index of the current hidden state, j is the forthcoming hidden state index, and τ is the duration of the forthcoming one respectively. The probability of the transition from hidden state 7 to hidden state 3 at time steps 28 to 29 is $a_{7,3}p_{3,8}$.

Note again that hidden states and their duration are not given *a priori* but obtained by unsupervised learning. More precisely, we first optimize parameters that determine transition, duration, and emission categorical distributions so that a tuned model gives higher likelihoods to targeted pieces, and then obtain the best hidden state sequence based on the model. Next, we give a detailed description of the architecture of neural networks used in the neural HSMM.

4.2 Architecture of Neural HSMM

Conventional HSMMs have categorical parameters as matrices of parameters. In contrast, neural HSMM equips neural network components as functions for calculating transition, duration, and emission distributions. As mentioned in the previous section, the role of each distribution is as follows.

- Transition distribution is the probability of transition from a hidden state to another; it corresponds to transition of chord categories.
- Duration distribution is the probability of duration of a hidden state; it corresponds to the duration of a chord category.
- Emission distribution is the probability for a hidden state to emit a particular token. In our case, this is the probability for a chord category to emit a particular pitch-class vector.

Besides these three distributions, a special case of transition distribution, *i.e.,* initial state distribution, is described in Section 4.2.2.

The same graphical representation (**Figure 3.3a**) applies to neural HSMMs, but the categorical distributions are obtained as outputs of neural networks that can employ additional musical contexts for the calculation. We show these networks and additional contexts in the following paragraphs: hidden state transition probability (Section 4.2.1), initial state probability (Section

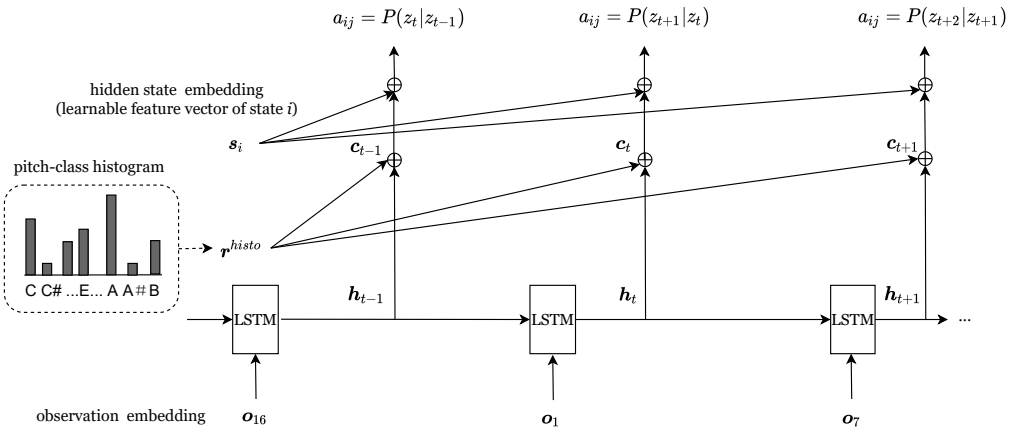


Figure 4.2: The network architecture for calculating transition probabilities a_{ij}. $\boldsymbol{s}_i$(s) are the set of hidden state embeddings. $\boldsymbol{r}^{histo}$ is an additional context of pitch-class histogram. $\boldsymbol{h}_t$ is another additional context of embedded feature of preceding observations by the LSTM. $\boldsymbol{o}_{16}$, $\boldsymbol{o}_1$, and $\boldsymbol{o}_7$, are observation embeddings associated with observed tokens: $x_{t-1} = 16$, $x_t = 1$, and $x_{t+1} = 7$.

4.2.2), duration probability (Section 4.2.3), and emission probability (Section 4.2.4)[2].

4.2.1 Hidden State Transition Probability

We denote a hidden state at time step t as z_t and state index of which as i or j. Given the number of hidden state S, each hidden state $i \in S$ has a transition distribution to the next hidden state $j \in S$, which is denoted as a_{ij}, and thus $\sum_j a_{ij} = 1$. Note that, in HSMM (Residential-time HMM), the self-transition a_{ii} is set to zero ($\rightarrow$ Section 3.3.2) since it is not the transition distribution but the duration distribution that manages the duration of a hidden state. Instead of having a_{ij} just as a single learnable value, we prepare a function that calculates it by employing neural network components as follows,

$$a_{ij} = P(z_{t+1} = j | z_t = i) = \text{softmax}_j(\text{MLP}_3([\boldsymbol{s}_i; \boldsymbol{c}_t])) \tag{4.1}$$

where MLP_3 is a 3-layer Multi Layer Perceptron with one input layer, two hidden layers, and one output layer. A hyperbolic tangent (tanh) activation function follows after each hidden layer ($\rightarrow$ Section 3.3.4). The output layer

[2]The summary of notations (**Table C.1**) and model settings (**Table C.2**) can be found in Appendix.

of this function is softmax (3.10) to satisfy the condition $\sum_j a_{ij} = 1$. The output layer size of the transition MLP_2 in (4.1) should be smaller by one than the number of states $(S-1)$ since it does not allow self-transitions. Since a_{ij} corresponds to the transition probability of hidden state i to j, the input for the network includes a feature for hidden state i. We provide a hidden state embedding $\boldsymbol{s}_i$, which is a learnable vector associated with hidden state index i. This hidden state embedding is also jointly used in the network for the duration and emission probability (described in Section 4.2.3 and Section 4.2.4); it would help capture relationships between hidden states (chord categories) and emissions (observed pitch-class vectors).

Taking further advantage of neural HMM, we introduce two additional contexts for calculating hidden state transition probability: a pitch-class histogram and an embedded feature of preceding pitch-class vectors. We feed a concatenation of these two contexts $\boldsymbol{c}_t$ to (4.1).

Additional context 1: Pitch-class histogram (**HISTO**)

The first additional context is a pitch-class histogram that is the frequency of occurrence of each pitch class calculated from an entire phrase, which represents tonality. We split a piece into chord sequences at each *fermata* (*point d'orgue*) that works as a full-stop marker of a lyric and then sum up the duration of each pitch-class to obtain a pitch-class histogram of a sequence. Even though the key signature would distinguish the main tonality of a piece[3], there would be local modulations. Thus we employ such a histogram as indirect information for local tonality. We obtain the feature of a pitch-class histogram of a chord sequence $\boldsymbol{r}^{histo}$ as follows.

$$\boldsymbol{r}^{histo} = \mathrm{MLP}_2(\boldsymbol{v}^{histo}) \tag{4.2}$$

where $\boldsymbol{v}^{histo} \in \mathbb{R}^{12}$ is the raw pitch-class histogram of a sequence.

Additional context 2: Embedded feature of preceding observations by the LSTM (**LSTM**)

The second additional context is an embedded feature of preceding observations by the Long-Short Term Memory (LSTM) [41], based on the idea from [80]. LSTM is a variant of Recurrent Neural Network (RNN) that recursively feeds observations; thus, it is suitable for a feature function for sequential data (→ Section 3.3.4). In our case, it

[3]However, the main tonalities of pieces that are written in church modes (*e.g., dorian*) would not correspond to the key signature system of modern tonality. In addition, the key signature alone does not distinguish major/minor. Furthermore, some minor pieces retain the feature of church modes [75] and frequently proceed relative major keys [73, 84].

corresponds to a feature of a temporal sequence (from $t = 1$ to the current time step t) of pitch-class vectors. At each time step, we input an observation embedding $\boldsymbol{o}_k$ that is associated with observation $\boldsymbol{v}_k^{pitch}$ at t[4] to LSTM as follows (4.3)(4.4).

$$\boldsymbol{o}_k = \tanh(\mathrm{MLP}_2(\boldsymbol{v}_k^{pitch})) \tag{4.3}$$

$$\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{o}_k, \boldsymbol{h}_{t-1}) \tag{4.4}$$

In the calculation of an observation embedding (4.3), $\boldsymbol{v}_k^{pitch}$ is a binary pitch-class vector, that is, a 12-dimensional vector of 1/0. For example, if the pitch-class vector is $(2, 5, 7, 11)$, the corresponding binary pitch-class vector is $(0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1)$. We expect that binary pitch-class vectors give raw information of observations instead of symbolized token indices, and it is also used in the emission distribution network described in Section 4.2.4.

Finally, we concatenate the two contexts and obtain the context vector as (4.5), where $[;]$ denotes vector concatenation. This context vector is applied to the transition probability network (4.1).

$$\boldsymbol{c}_t = [\boldsymbol{r}^{histo}; \boldsymbol{h}_t] \tag{4.5}$$

We illustrate the network for calculating transition probability a_{ij} in **Figure 4.2**, which we have described in this section. Note that transition probability thus dynamically varies by contexts in neural HSMM, though it is static in conventional models. Therefore, while we train a model without distinguishing major/minor pieces or local modulations, we expect that the additional contexts adjust transition probabilities automatically.

4.2.2 Initial Hidden State Probability

Since the first pitch-class vector at the time 1 does not have a preceding observation, this special distribution for the initial hidden state is provided. We set an external initial hidden state z_0 at time step 0 that has no emission and duration 1 to meet the initial boundary condition of HSMM. The initial hidden state probability ρ_i is calculated as follows. ρ_i is the only component that remains almost the same as the conventional model, except that it is normalized by the softmax function.

$$\rho_i = P(z_0 = i) = \mathrm{softmax}_i(\boldsymbol{\pi})$$

where i is a hidden state index, $\boldsymbol{\pi} \in \mathbb{R}^S$ is the learnable weight vector, and S is the number of hidden states.

[4]In this example, $x_t = k$, which means that the token index for x_t is k.

4.2.3 Duration Probability

The duration probability determines how long the hidden state i (*i.e.,* a chord category) will reside in the same hidden state. The network for calculating it is as follows.

$$p_{i\tau} = \mathrm{softmax}_\tau(\mathrm{MLP}_3([\boldsymbol{s}_i; \boldsymbol{r}_t^{beat}])) \tag{4.6}$$

$$\boldsymbol{r}_t^{beat} = \mathrm{MLP}_2([v^{timesig}; v_t^{beat}]) \tag{4.7}$$

Additional context 3: Beat position (**BEAT**)

In (4.6), $\boldsymbol{r}_t^{beat}$ is an additional context for the calculation of duration probability, which is expected to give metrical information. $[v^{timesig}; v_t^{beat}]$ is a 2-dimensional vector that consists of a beat position and the numerator of a time signature. In this study, $v^{timesig}$ is constant ($= 4$) since we only treat four-four time (4/4) pieces, while v_t^{beat} could vary among $\{1.0, 1.25, \cdots, 4.5, 4.75\}$.

The output layer size of the duration MLP_2 in (4.6) is 16, a maximum duration length corresponding to a whole note. The hidden state embedding $\boldsymbol{s}_i$ used in the network for transition probability is used again and is expected to help the model jointly learn the transition and duration distribution of hidden state i.

4.2.4 Emission Probability

We employ a discrete HSMM in this study; therefore, each observation is associated with token index $k \in V$ in the vocabulary, as mentioned in the example in Section 4.1. The probability that hidden state i yields a token index k (b_{ik}) is calculated as follows, based on [80].

$$b_{ik} = P(x_t = k | z_t = i) = \mathrm{softmax}_k(\boldsymbol{s}_i^\mathsf{T} \boldsymbol{o}_k + l_k) = \frac{\exp\left(\boldsymbol{s}_i^\mathsf{T} \boldsymbol{o}_k + l_k\right)}{\sum_{k'} \exp\left(\boldsymbol{s}_i^\mathsf{T} \boldsymbol{o}_{k'} + l_{k'}\right)} \tag{4.8}$$

Additional context 4: Observation embedding from binary pitch-class vector (**PITCH**)

We use the same observation embedding $\boldsymbol{o}_k$ used in the transition probability (4.3) calculation for emission probability (4.8). Instead of employing weight matrices or MLPs, the emission probability is learned as the dot product of a hidden state embedding and an observation embedding. The bias value $l_k \in \mathbb{R}$ would help to consider the frequency of each chord. In contrast to the conventional discrete HMM that cannot use the raw information about the observation once converted into a token index, our model does not lose it by the observation embedding.

4.3 Training

It is known that there is no analytical solution to obtain an optimal parameterization of an HMM that maximizes the likelihood of the observed sequence [68]. Although the widely used Baum-Welch re-estimation algorithm (a kind of the Expectation maximization (EM) algorithm) is able to obtain a locally optimal parameterization, it would be stuck in bad local optima when the likelihood surface is complex [68]. On the other hand, we can utilize gradient-based methods to maximize the marginal likelihood $\sum_{n=1}^{N} \ln P(\mathbf{x}_{1:T}^{(n)})$ where $\mathbf{x}_{1:T}^{(n)}$ is an observed sequence since it is an optimization problem [68, 48, 54]. We employ the known dynamic programming named the forward algorithm for HSMM to calculate the marginal likelihood [68, 90] ($\rightarrow$ Section 3.3.2).

Neural network models are generally trained by gradient-based optimizers and the backpropagation for gradient computing. Recently, along with the success of neural networks, efficient optimizers also have been proposed. We can naturally utilize gradient-based optimization when implementing the HSMM as a neural network [48]. In this study, we use one of the latest optimizers, RAdam [55], which adjusts the learning rate automatically.

We empirically found the effectiveness of the gradient-based method by comparing neural HSMMs of minimum architecture (we call them *baseline* models) with conventional models trained by the EM algorithm; the experiments are shown in Appendix A.

4.4 Experimental Setups

Table 4.2: The statistics of the dataset.

	#pieces	(maj.)	(min.)	(dor.)	(other)	#phrases	#obs. tokens
Train	185	(89)	(70)	(23)	(3)	1155	38452
Eval.	51	(21)	(23)	(6)	(1)	314	10408
Test	54	(28)	(20)	(5)	(1)	350	11868

We use J.S.Bach's four-part chorales by the Riemenschneider numbering system (1-371) from the Music21 Corpus [21] as our dataset resource, details of which were described in Section 3.2. We only contain four-four time (4/4) pieces and exclude 27 pieces that are not four-part voices or have some problems, such as a collapsed format. Thus, the number of pieces is 290. We regard each phrase as an independent sequence, however, when we provide

these phrases to training, evaluation, and testing sets, we give randomness over pieces, not phrases.

In this study, we take over the assumption of transpositional equivalency between keys, as mentioned at the beginning of this chapter. This is basically equivalent to transposing all the major pieces to **C** major and all the minor ones **a** minor, relying on key signatures. However, some pieces of the corpus are written in *dorian* scale without any key signature. From the viewpoint of modern tonality, we regard them in **d** minor; however, we do not shift them to **a** minor. Also, local modulations are shifted in the same way, so as to preserve the relative positions of constituent notes. Finally, we ignore the difference by octaves in pitch events and inversion in chords.

We assign token indices (k) for pitch classes, the accumulated pitch-duration of them in the dataset over 95%, and then the remaining chords are merged to "Others." The obtained vocabulary size was 80, including "Rest" (**Table 4.1**).

In training, we set the mini-batch 8. We train models up to 500 epochs; however, the process is stopped when the lowest loss on the evaluation set is not updated over 20 epochs. We apply the dropout [40] to each hidden layer of MLPs, which randomly ignores some ratio of neurons (12.5% in our case) during training to avoid over-fitting.

4.5 Evaluation by Perplexity

Perplexity (Section 3.3.3) is a widely used metric for sequential modeling without any reference data [5, 15, 16, 47, 82]. It corresponds to the inverse probability normalized by the sequence length and thus is an indicator of the model's generalization performance.

We examine the performance on multiple numbers of hidden states: 3 to 16. We conduct the experiments with three different random seeds of $\{0, 1, 2\}$ for each number of hidden states and report the averaged score.

In addition to the neural HSMM, we implement a baseline model that represents probabilities as simply learnable weight vectors or matrices with softmax output layers. The baseline model is almost "Non-Neural" but an HSMM tuned by the same gradient-based optimizer. We compare the proposed model with this baseline to see how efficiently the elaborated neural network components work. In **Figure 4.3**, we can see that the elaborated neural models considerably outperformed the baselines.

The ablation study also shows the efficacy of additional contexts, as shown in **Table 4.3**. Removing any additional contexts degraded the perplexity: the pitch histograms ($-$**HISTO**), an embedded feature of preceding obser-

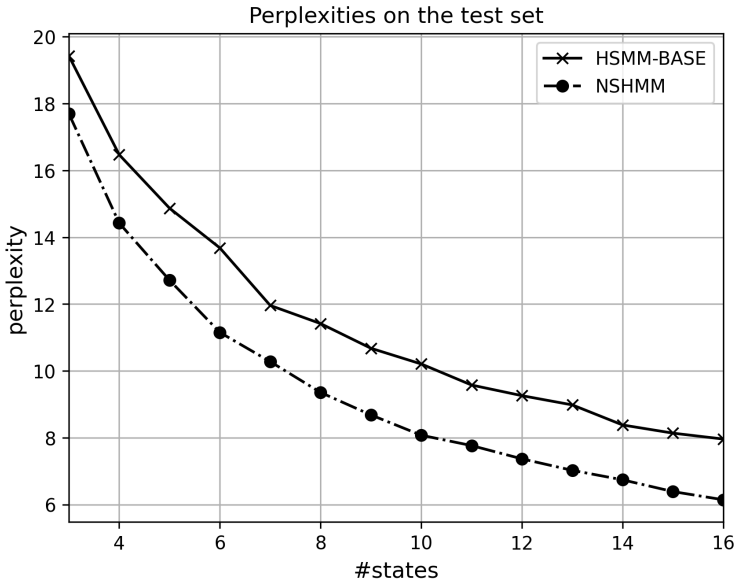


Figure 4.3: Averaged perplexities by three trials with random seeds of $\{0, 1, 2\}$ on the testing set.

vations by the LSTM (−**LSTM**), the beat positions (−**BEAT**), and the observation embeddings from binary pitch-class vectors (−**PITCH**). Among these contexts, removing[5] the observation representation by pitch classes (−**PITCH**) led to a significant drop. Since we do not employ MLPs for the architecture of calculation of emission probability but force a model to learn relationships between hidden states and vocabularies directly, the informa-

[5]In this case, we used an observation embedding of learnable weights in the same way as the hidden state embedding instead of the pitch-class one.

Table 4.3: Ablation studies. Averaged perplexities by three trials with random seeds of $\{0, 1, 2\}$ on the testing set. The bold numbers are the best score in the same number of the hidden state.

#hidden states	4	8	16	32
BASE	16.47	11.42	7.96	5.64
NHSMM	**14.43**	**9.36**	**6.14**	**4.74**
−**HISTO**	15.18	9.78	6.51	4.92
−**LSTM**	14.94	9.67	6.47	5.07
−**BEAT**	14.85	9.65	6.29	4.77
−**PITCH**	16.88	10.64	7.23	5.38

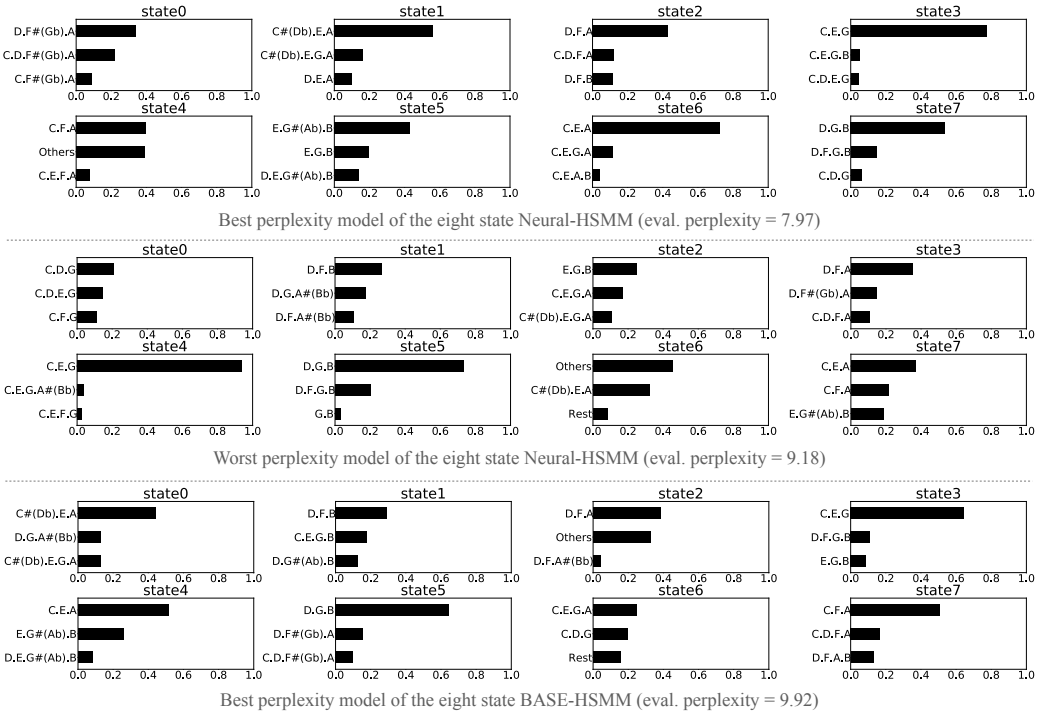


Figure 4.4: Comparison between (Top) neural HSMM of the best evaluation perplexity among the three random seeds, (Middle) neural HSMM of the worst evaluation perplexity, and (Bottom) base HSMM of the best evaluation perplexity. The bar charts show the top three emissions per each hidden state.

tion of raw pitch classes would help the learning.

4.6 Qualitative Analysis for Induced Clusters

This section presents a qualitative analysis and discusses the induced chord clusters. We focus on the eight-state models since we have transposed pieces not to have key signatures and have assumed that each cluster would roughly correspond to a triad on the diatonic scale or otherwise to the rests.

4.6.1 Induced Clusters and Model's Perplexities

We investigate if we can find clearer clusters in the higher-scored model in obtained examples. **Figure 4.4** shows the emission probabilities of the best neural HSMM, the worst neural HSMM, and the best baseline model. Note that we have executed three trials of training for each model with a

Table 4.4: Chord categories obtained by the best scored 8-states neural HSMM. The chord category is named after the chord name of the top emission for each hidden state.

hidden state index		state0	state1	state2	state3
chord category		$\hat{\text{D}}$	$\hat{\text{A}}$	$\hat{\text{d}}$	$\hat{\text{C}}$
emit. top3	(1)	D.F♯(G♭).A	C♯(D♭).E.A	D.F.A	C.E.G
	(2)	C.D.F♯(G♭).A	C♯(D♭).E.G.A	C.D.F.A	C.E.G.B
	(3)	C.F♯(G♭).A	D.E.A	D.F.B	C.D.E.G
hidden state index		state4	state5	state6	state7
chord category		$\hat{\text{F}}$	$\hat{\text{E}}$	$\hat{\text{a}}$	$\hat{\text{G}}$
emit. top3	(1)	C.F.A	E.G♯(A♭).B	C.E.A	D.G.B
	(2)	Others	E.G.B	C.E.G.A	D.F.G.B
	(3)	C.E.F.A	D.F.G♯(A♭).B	C.E.A.B	C.D.G

different random seed among $\{0, 1, 2\}$. Here the "best" model means that the evaluation perplexity of which is the smallest among the three trials.

We observe that the clusters obtained by the best scored neural HMM mainly consist of the chords on diatonic scales (C major and a minor) and its commonly-used borrowed chords such as D major chord. **Figure 4.4** shows the emission probabilities of the top three emissions for each hidden state index. For ease of readability, we name chord categories by the name of the most frequent output chord for each hidden state index, as summarized in **Table 4.4**.

Although the most frequent chord is representative of the category, it does not correspond to a unique chord but a set of emission probabilities; therefore, we add "hat" to it. The cluster $\hat{\text{A}}$ may emerge from two reasons: the common use of the *Picardy* third or the dominant of *dorian* mode pieces. It is worth noting that seventh and passing chords are merged into appropriate clusters, *e.g.*, C.D.E.G in the same hidden state for C.E.G (state3) and D.F.G.B for D.G.B (state7).

Even though employing the same neural HSMM, the worst perplexity model (in the middle of **Figure 4.4**) seems less appropriate than the best model. For example, C.E.A is mixed up with C.F.A and E.G♯(A♭).B in state7, and C.E.G (state4) and passing chords around it (state0) are separated. Although we would be able to choose a model by perplexity, we admit that difficulty in obtaining the global optimum still exists even though we employ the efficient gradient-based optimizer.

The best baseline model (the evaluation perplexity of which is 9.92) is still worse than the worst neural HSMM (9.18). Moreover, not only the best

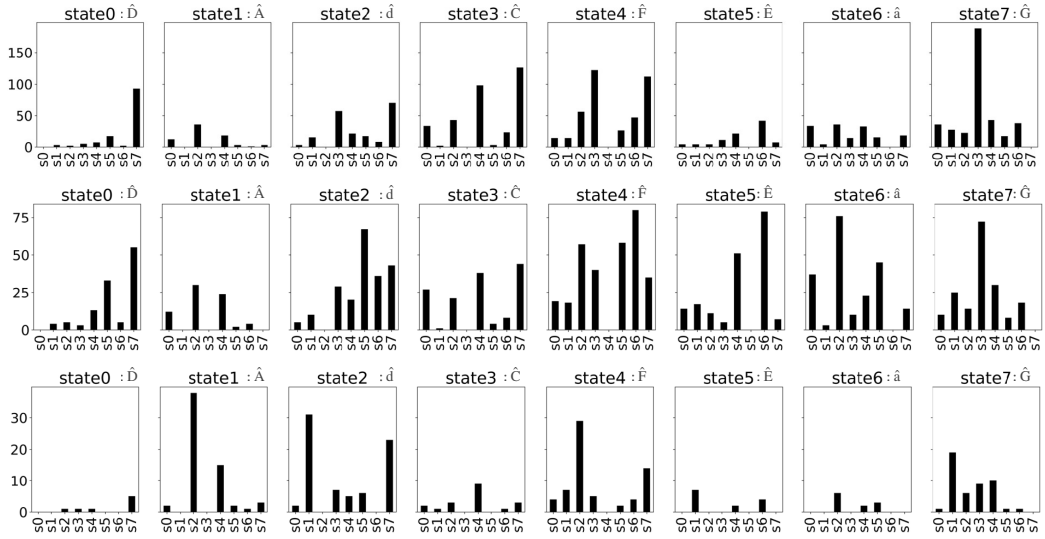


Figure 4.5: Counts of hidden state transitions: (Top) major pieces, (Middle) minor pieces, and (Bottom) *dorian* pieces. The sequence of hidden states is calculated by the Viterbi algorithm.

baseline model scored a worse perplexity, but it possessed more miscellaneous clusters than neural HSMMs. However, we can observe that C.E.G and D.F.G.B of state3, or D.G.B and D.F♯(G♭).A of state5, in the same tonality, would tend to be merged into the same category in the baseline HSMM.

4.6.2 Hidden State Transitions

Since chord functions lie in the regularity of chord progressions, we try to find them by investigating the transitions between obtained chord categories. We count the hidden state transitions on major pieces, minor pieces, and *dorian* pieces separately to see whether the model appropriately reflects the difference of tonalities in its state transitions. Note that, instead of examining the hidden state transition probability directly, we count the number of transitions after decoding the sequence of hidden states by the Viterbi algorithm[28] since the hidden state transition probability changes by context.

We show the result of hidden state transition properties by the best scored neural HSMM in **Figure 4.5** (the emission probability of which is shown at the top of **Figure 4.4**). In major pieces, the tendency that {state0:D̂, state2:d̂, state4:F̂} (secondary dominant or subdominant) → state7:Ĝ (dominant) → state3:Ĉ (tonic) is noticeable. While in minor pieces, the tendency described above suggests the existence of the relative major keys, which is

consistent with previous works [73, 84]. In addition, a strong transition from state5:Ê (dominant) → state6:â (tonic) are observed. Unlike in major pieces, the transition from state2:d̂ (subdominant) → state5:Ê (dominant) is increased in minor pieces; it corresponds to the transition of subdominant → dominant. Finally, in *dorian* pieces, transitions from state1:Â (dominant) → state2:d̂ (tonic) are observed. Unlike **C** major and **a** minor pieces, the transition from state7:Ĝ no longer has the strong tendency of proceeding to state3:Ĉ but tends to proceed to state1:Â instead, corresponding to the progression of subdominant → dominant.

4.7 Discussion on an Analysis by the Model

In this section, we discuss the adequacy of our model from an obtained result. We select BWV267 from our testing set since a human analysis on which is publicly available [21][6]. Although this study is not aimed to reproduce the human analysis, we consult it as a possible interpretation for local modulations and chord annotations. Note that we normalized the score to have no key signature; thus, the main key became **C** major, while the original key was **G** major. We show extractions of the result in **Figure 4.6**[7]. According to the human analysis [21], there are local modulations to {**F** major, **G** major, **d** minor, **g** minor} keys in this piece.

In this example, we could see that the model found an effective set of clusters to cover chords appeared that in a piece, including local modulations. For example, in the section of the **G** major key (phrase No.4, time step 21–32), we observed the cluster of D̂ (state0), which was a borrowed chord seen from the **C** major key. This cluster D̂ was used again in the section of the **g** minor key (phrase No.7, time step 7–32). Interestingly, g minor chords, *i.e.,* D.G.A♯(B♭) ($k = 21$) and D.F.G.A♯(B♭) ($k = 60$), were classified into the same cluster as Ĝ (state7) since g minor chord and G major one shared the dominant (D) (state0: D̂). Similarly, in the section of **F** major (phrase No.4, time step 1–20), C major dominant seventh chords, *i.e.,* C.E.G.A♯(B♭) ($k = 24$), were classified into the same cluster as C major chords (state3: Ĉ). According to Schoenberg, local modulations basically remain in closely related keys and thus can be analyzed by altered chords [75]. In the case of g/G chords and C/C7 chords described above, these alternations (g or C7) do not change the probable progression of chords and thus would be classified

[6] https://github.com/cuthbertLab/music21/tree/master/music21/corpus/bach/choraleAnalyses

[7] The result for the whole piece is shown in Appendix: **Figure D.1** (phrase No.1–4) and **Figure D.2** (phrase No.5–8).

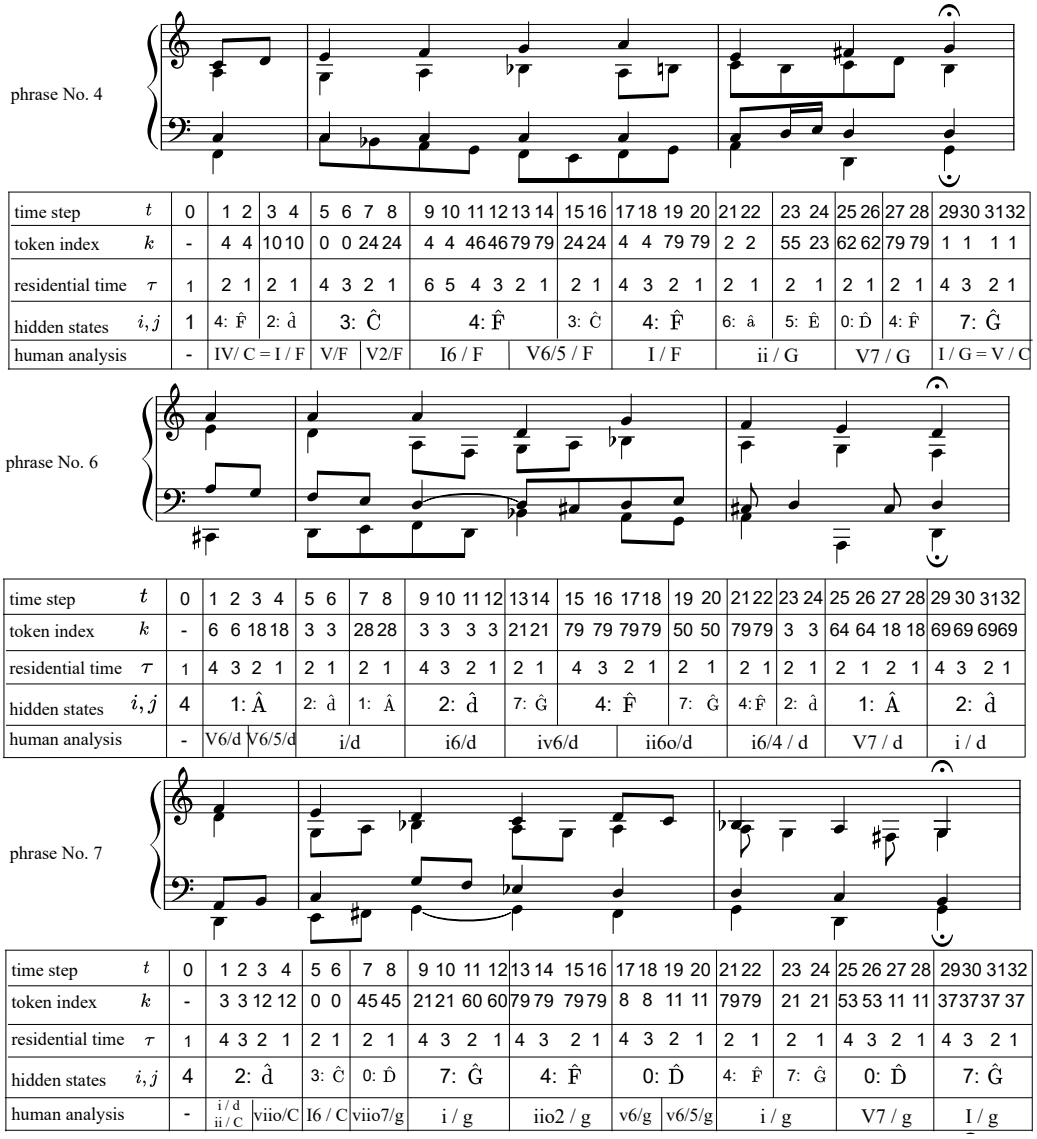


Figure 4.6: Chord classification by the neural HSMM on BWV267 (Excerpt)

into the same function as the one on the diatonic scale (G or C).

In the longest modulation section of **d** minor (phrase No.6), we could see that the cluster of the tonic chord (state3: $\hat{\text{C}}$) had not appeared and that the cluster of the d minor chord (state2:$\hat{\text{d}}$) tended to proceed to the A major chords (state1:$\hat{\text{A}}$).

Since we ignored inversions of chords in this study, we basically could not capture the difference between them. We admit that it needs further consideration and left it as future work. In addition, we observed more "Others" tokens ($k = 79$) in modulation sections that hindered the analysis. "Others" or "Unknown" token has been introduced in discrete HMMs to avoid useless increase of dimension of the emission distribution. However, once an observation is converted into "Others", the raw information about it (which is pitch-class vector in our case) will be lost. More effective treatment for less frequent observations or more direct representation for emission distribution is another important future work for us.

4.8 Chapter Summary

In this chapter, we introduced the neural hidden semi-Markov model (HSMM) that automatically segments surface pitch classes and classifies them into chord categories. In contrast to the previous works [44, 73, 84, 88], we did not distinguish tonalities (major, minor, and other tonalities in *church modes*) to train the model. We did so for both conceptual and practical reasons. For a conceptual reason, chords, functions, and tonalities are mutually dependent [70] and should not be analyzed independently. In practice, we cannot easily distinguish tonalities (especially local modulations) in advance when we consider applying the model to more complicated pieces such as ones in the post-romantic period. Hence, we proposed a model that allows for chord segmentation and classification while considering tonality as a context, without distinguishing it in advance. We adopt the two techniques in order to achieve the above.

- We employed hidden semi-Markov model (HSMM) instead of HMM, which introduced the notion of the duration of a hidden state.
- Neural networks are utilized for calculating categorical distributions to embed the pitch-class distribution, preceding chord sequences, beat information, and so on, as additional contexts.

The experimental results satisfactorily showed the efficacy of such contexts in terms of *perplexity* in comparison with the conventional HSMM and

other ablation studies (Section 4.5). Furthermore, we successfully obtained clear chord clusters with the best scored proposed model where seventh chords and borrowed chords were classified into appropriate clusters (Section 4.6.1). Also, we have shown a full exemplification of BWV267 (Section 4.7).

We found that the transition between hidden states well simulated the known functional harmony theory. We examined it by counting the number of hidden state transitions on major pieces, minor pieces, and *dorian* pieces separately. As a result, we confirmed that the distribution of hidden state transitions was consistent with the known classical theory (Section 4.6.2). However, it is a major drawback of the proposed model that cannot explicitly distinguish the difference of tonalities but relies on the pre-defined annotation of them in evaluation. Therefore, we further extend the model in order to incorporate unsupervised tonality detection, as described in the next chapter.

Chapter 5

Unsupervised Clustering of Tonality

In the previous chapter (Chapter 4), we introduced the neural hidden semi-Markov model (HSMM) that automatically identified a chord sequence from a surface musical structure; it automatically segmented and classified surface pitch classes into chords without relying on conventional chord symbols. In addition, we found that the model appropriately reflected the difference in tonalities in its hidden state transitions.

Although the neural HSMM automatically adjusts hidden state transition probability with the help of efficient additional contexts, it does not explicitly detect changes in the trend of transition probability. Therefore, we counted the transition of classified chords for major, minor, and *dorian* pieces separately. In this chapter, we further extend the model by employing a *teacher-student* architecture to classify tonality[1].

We propose a methodology to obtain a set of tonalities that appeared in targeted pieces by classifying chord transitions, which is an extension of the neural hidden semi-Markov model (Chapter 4). We employ it as a *teacher* model that classifies surface pitch-class vectors into chord categories and gives counts of transitions between chords. We also prepare *student* models that simplify the network for calculating transition probability by replacing it with a set of learnable matrices just like the conventional hidden semi-Markov model (HSMM). The model equips multiple students (*i.e.,* transition matrices), which are expected to represent different tonalities. Note that the teacher model does not know the difference between tonalities, but students discover it by the learning. In this sense, students are not just simplified versions of the teacher model.

[1]See footnote 2 in Chapter 1

Before introducing the proposed model, we show a preliminary experiment that suggests the neural HSMM intrinsically represents similarities between tonalities in the next section (Section 5.1). After that, we describe the proposed teacher-student model for unsupervised mode classification in Section 5.2. Then, we show experimental setups (Section 5.3) and results (Section 5.4).

5.1 Tonality Distance by KL Divergence; a Preliminary Experiment

According to Schoenberg, at least one chord is shared by the two keys when a key changes to another (modulation) [75]. In other words, modulations tend to be conducted between keys that share more common pitches. Therefore, a cadential progression indicates a particular key, even when most pitches are common [70, 75].

As a generative statistical model, a trained neural HSMM gives the marginal likelihood $\ln P(\mathbf{x}_{1:T}; \boldsymbol{\theta})$[2] to an observed sequence $\mathbf{x}_{1:T}$. We expect that a model trained with a dataset of a particular key gives the highest likelihood to the targeted key and higher likelihoods to closely related keys. Therefore, the difference between the marginal likelihoods calculated by two different keys is expected to simulate the distance between the two keys.

The notion above is formally expressed as the Kullback-Leibler divergence (KL divergence) [46, 52, 76][3].

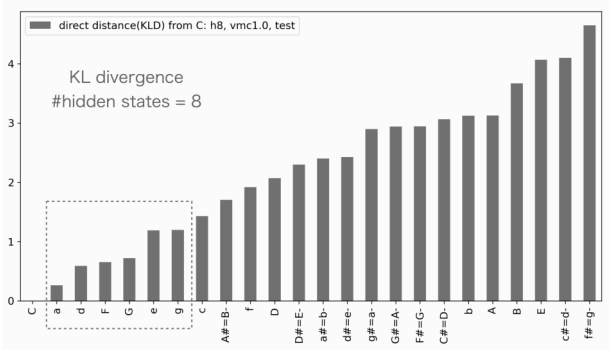
$$D_{KL}(\boldsymbol{\theta}_{\mathsf{key1}}, \boldsymbol{\theta}_{\mathsf{key2}}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} [\log P(\mathbf{x}^{\mathsf{key1}} | \boldsymbol{\theta}_{\mathsf{key1}}) - \log P(\mathbf{x}^{\mathsf{key1}} | \boldsymbol{\theta}_{\mathsf{key2}})]. \quad (5.1)$$

In equation (5.1), $\mathbf{x} = \mathbf{x}_{1:T_i}$ is an observed sequence with length T_i, N is the number of observed sequences, and $\boldsymbol{\theta}_{\mathsf{key1}}$ and $\boldsymbol{\theta}_{\mathsf{key2}}$ are parameters of HMMs tuned by different keys. The superscript $\mathbf{x}^{\mathsf{key1}}$ represents that the data $\mathbf{x}^{\mathsf{key1}}$ is expected to follow the same distribution as the model $\boldsymbol{\theta}_{\mathsf{key1}}$. To apply this to an estimation of tonality distance, we regard $\boldsymbol{\theta}_{\mathsf{key1}}$ as a model tuned by data of $\mathsf{key1}$ and $\boldsymbol{\theta}_{\mathsf{key2}}$ by $\mathsf{key2}$.

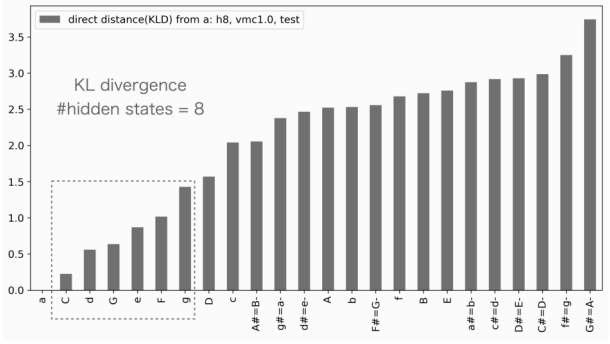
Here, we once put the main objective of unsupervised recognition of tonality aside and examine an intrinsic property for the model to express the tonality distance by employing the KL divergence, as a preliminary experiment. Different from the previous chapter's model (Chapter 4), we omitted

[2] $\boldsymbol{\theta}$ is the set of parameters of the model.

[3] (5.1) is an approximated form of KL divergence specialized in HMM.



(a) KL divergences between **C** major and each of 24 keys.



(b) KL divergences between **a** minor and each of 24 keys.

Figure 5.1: Tonality distance, *i.e.,* KL divergences, based on the 8-states neural HSMMs.

the pitch-class histogram (4.2) from the additional contexts since it might be an unfair advantage to the prediction of tonality. In this preliminary experiment, we trained **C** major and **a** minor models separately so that we could compute KL divergences between major keys and minor ones[4][5].

The obtained result when key1 was fixed to **C** major is shown in **Figure 5.1a**. We arranged keys in ascending order of the KL divergence. Although it would not be surprising that **a**, **F**, and **G** had quite small distances,

[4]Although we divided pieces to **C** major or **a** minor by their main tonalities, here again, we left local modulations untouched. Therefore, related keys which appeared as local modulations were expected to have smaller KL divergence

[5]Also, we reduced the assignment of the "Others" token to avoid the unexpected increase in its ratio when we included those between remote keys for this preliminary experiment. In particular, we change the proportion of the "Others" token to 0.2%, which means we give a unique token index to a pitch-class vector when the accumulated duration of which is larger than a quarter note.

Table 5.1: Counts and proportions of local modulations from the professionally annotated analysis of 20 pieces [21]. The main tonalities are transposed to **C** major or **a** minor.

C major			A minor		
key	count	prop.	key	count	prop.
C	36	50.7	a	30	43.5
G	14	19.7	C	16	23.2
d	8	11.3	G	11	15.9
a	6	8.5	e	6	8.7
F	6	8.5	d	5	7.2
g	1	1.4	D	1	1.4
Total	71	100	Total	69	100

the closeness of **d** would be evidence that the obtained result faithfully reflected the targeted pieces. The result is consistent with the professional analyses for the first 20 pieces (by Riemenschneider numbering) that are publicly available in the Music21 corpus [21][6], where we could see a considerable amount of modulation from **C** to **d**, as summarized in **Table 5.1**. Furthermore, **Table 5.1** shows all the local modulations that appeared in the 20 pieces. Thus, the set of related keys from professionally annotated local modulations: {**G**, **d**, **a**, **F**, **g**} was completely included in the closest six keys in our result. Similarly, the result for the case of **a** minor showed considerable agreement with the analysis, as shown in **Figure 5.1b**.

5.2 Unsupervised Clustering of Tonality by Teacher-Student Architecture

5.2.1 Framework

In the previous section (Section 5.1), we have seen that the neural HSMM could assess the similarities between tonalities. Based on the discussion by Riemann that a chord changes its function in modulation [70], we expect that the difference in tonality mainly appears in hidden state transitions[7].

Although we have examined the difference by separating the pieces by keys so far, we now classify chord transitions in an unsupervised manner and

[6] → footnote 6 in Chapter 4

[7] Here, we normalize pieces so as not to have key signatures and assume that modulations are mainly within closely related keys.

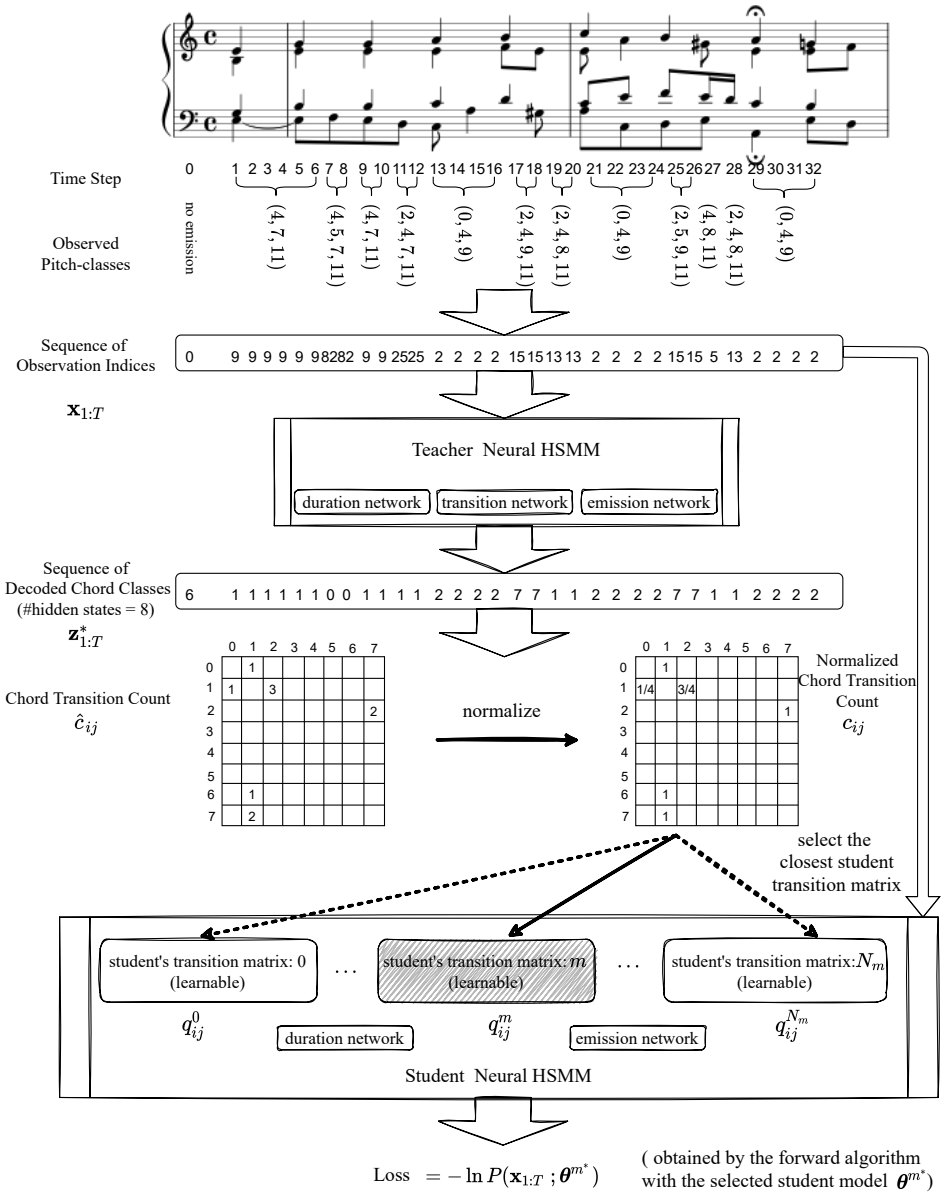


Figure 5.2: Illustration for framework of proposed teacher-student architecture.

obtain categories of tonality. In order to achieve this, we extend the neural HSMM with *teacher-student* architecture, as illustrated in **Figure 5.2**. The learning procedure is summarized as follows.

1. Obtain a sequence of hidden states (*i.e.,* chord categories) from the *teacher* model.

2. Count transitions between chord categories where we do not include self-transitions. We normalize the obtained count matrix $M_{count} = (c_{ij})$ so as to meet $\sum_j c_{ij} = 1$.

3. Select the closest matrix in the student model where the similarity is calculated as Frobenius inner product of count matrix (c_{ij}) and *student* hidden state transition matrix (q_{ij}): $\sum_{ij} c_{ij} q_{ij}$.

4. Calculate the marginal likelihood $\ln P(\mathbf{x}_{1:T}, \boldsymbol{\theta}^{m^*})$ (where $\mathbf{x}_{1:T}$ is a sequence of observation indices) with the selected student's model $\boldsymbol{\theta}^{m^*}$ and optimize it with a gradient-based optimizer, where model parameters except for the transition matrix are shared with the teacher model and fixed.

5.2.2 Teacher Model

We basically follow the implementation of neural HSMM described in the previous chapter (Chapter 4). We use the eight hidden states model that was found to represent chords on diatonic scales and frequently borrowed chords. However, we remove the additional context of pitch-class histograms that ought to be acquired after the analysis of the entire phrase in order to focus on a purely transition-based model. Then, the neural network architecture for hidden state transition probability is modified as follows:

$$
\begin{aligned}
a_{ij} &= \mathrm{softmax}_j(\mathrm{MLP}([\boldsymbol{s}_i; \boldsymbol{h}_t])) \\
\boldsymbol{o}_k &= \tanh(\mathrm{MLP}(\boldsymbol{v}_k^{pitch})) \\
\boldsymbol{h}_t &= \mathrm{LSTM}(\boldsymbol{o}_k, \boldsymbol{h}_{t-1})
\end{aligned}
\tag{5.2}
$$

where a_{ij} is transition probability, i, j are indices of hidden states, $\boldsymbol{s}_i$ is a learnable feature of hidden states, and $\boldsymbol{h}_t$ is an additional context of the embedded feature of preceding observations by Long-Short Term Memory (LSTM). $\boldsymbol{o}_k$ is an observation embedding that is obtained from the corresponding pitch-class vector $\boldsymbol{v}_k^{pitch}$.

Given a sequence of observed pitch classes $\mathbf{x}_{1:T}$, we extract the sequence of chord categories (*i.e.,* the sequence of best hidden states) $\mathbf{z}^*_{1:T}$ with the Viterbi

algorithm [28] based on the teacher model. Thereafter, we count hidden state transitions from the sequence of the best hidden states and obtain the count matrix $M_{count} = (c_{ij})$, and then normalize it to meet $\sum_j c_{ij} = 1$. Note that we do not include self-transitions when we count the hidden state transitions; thus, the count matrix would be sparse, as illustrated in **Figure 5.2**.

5.2.3 Student Model

The student model simplifies the neural network for transition probability (5.2) to matrices; we denote it q_{ij}^m, where $m \in N_m$, and N_m is the number of students. The student transition matrices are implemented as learnable vectors and thus optimized by learning to become prototypes for tonalities.

Classification is conducted by selecting the student matrix $q_{ij}^{m^*}$ that has the highest similarity to the count matrix received from the teacher model. The similarity is simply calculated as the Frobenius inner product between the student's transition matrix q_{ij}^m and the count matrix c_{ij}: $\sum_{ij} c_{ij} q_{ij}^m$.

The teacher model is pre-trained, and parameters except for hidden state transition networks (5.2) are shared with the student model. When training, we calculate the marginal likelihood $\ln P(\mathbf{x}_{1:T}; \boldsymbol{\theta}^{m^*})$ with the selected student model $\boldsymbol{\theta}^{m^*}$[8] and optimize it with a gradient-based optimizer[9].

5.3 Experimental Setups

Table 5.2: The statistics of dataset.

	train	dev.	test
# piece	176	49	65
# phrase	1100	301	418

We use the same dataset in Chapter 4, which consists of J. S. Bach's four-part chorales from the Music21 Corpus [21][10]. Again, we randomly divide pieces into training, development, and testing sets. In addition, we specially reserve the first 20 pieces in the Riemenschneider numbering for testing since we use the human annotations for them [21] as *gold* data. Then, we split a piece into phrases at each *fermata* that indicates the end of a lyric paragraph in chorales and regard each phrase as an independent sequence. The statistics

[8] More precisely, the student model with the selected transition matrix $a_{ij} = q_{ij}^{m^*}$.

[9] We use the same optimizer (RAdam [55]) used in Chapter 4

[10] See also Section 3.2

of the dataset are shown in **Table 5.2**. Here, we only use four-four time (4/4) pieces and exclude pieces that are not four-part voices or have some problems, such as a collapsed format. Thus, we have 13 testing pieces with the human analysis.

As in Chapter 4, we normalize pieces so as not to have key signatures. While some pieces are not written in the same system of key signatures as the modern tonal system since they retain the feature of the *church modes*, we do not distinguish them beforehand by our human judgment. We transpose keys in the human analysis [21] in the same manner. The statistics of keys after normalization in the human analysis for the 13 pieces are shown in **Table 5.3**.

Table 5.3: The statistics of keys in the human analysis [21]. We regard sixteenth notes as one time step. Pieces are pre-transposed so that they have no key signature.

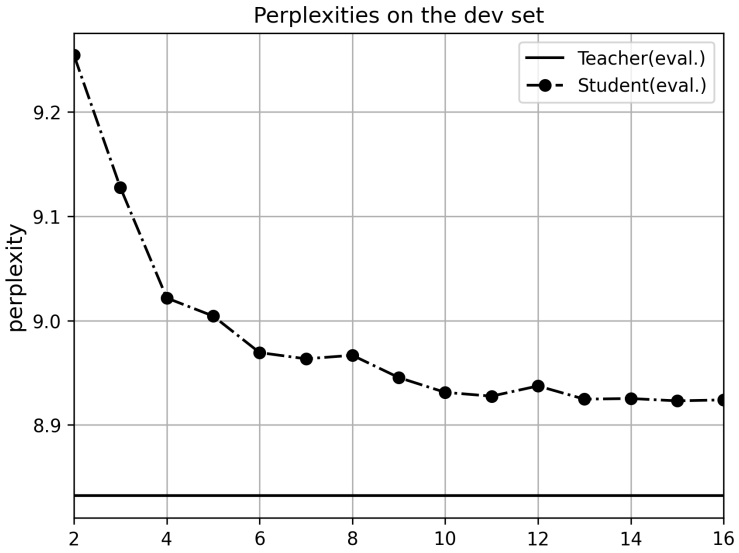
C	F	G	a	d	e	g
1196	176	360	570	598	8	66

We use the same model settings (**Table C.2**) and learning setups (*e.g.,* training epochs, the mini-batch size, and the dropout setting), as described in Section 4.4. Token indices are basically assigned in the same manner as Section 4.4; however, we slightly modify them so as to include pitch classes the accumulated duration of which is larger than four whole notes. Thus, the vocabulary is almost the same as **Table 4.1**, but three tokens are added: $(4, 6, 9) : (\mathrm{E}, \mathrm{F}\sharp(\mathrm{E}\flat), \mathrm{A})$, $(2, 4, 5, 11) : (\mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{B})$, $(1, 7, 9) : \mathrm{C}\sharp(\mathrm{D}\flat), \mathrm{G}, \mathrm{A}$.

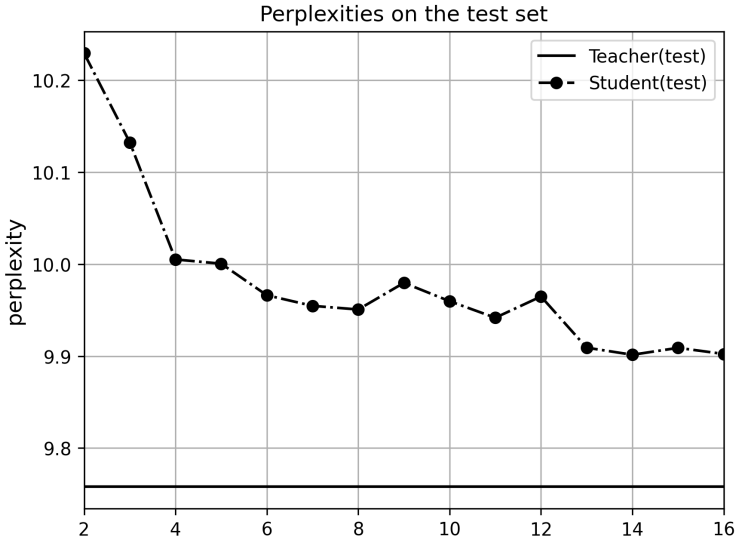
5.4 Results and Discussion

5.4.1 Evaluation by Perplexity

As described in Section 3.3.3 and Section 4.5, *perplexity* is a common evaluation metric that represents the model's generalization performance. However, previous works found that models with a larger number of parameters (*e.g.,* the number of hidden states) scored smaller (better) perplexity and thus hardly discovered an optimal setting of hyperparameters only by consulting perplexity [82, 85]. Therefore, we varied the number of student matrices (*i.e.,* cluster of tonalities) from 2 – 16. We expect that an appropriate number of clusters exists in this range; the modern tonal system has two *modes* (major scale and minor scale, or *ionian* and aeolian), while the medieval church modes are often categorized in 8 – 12 modes. In addition, local modulations



(a) Perplexities on the development set.

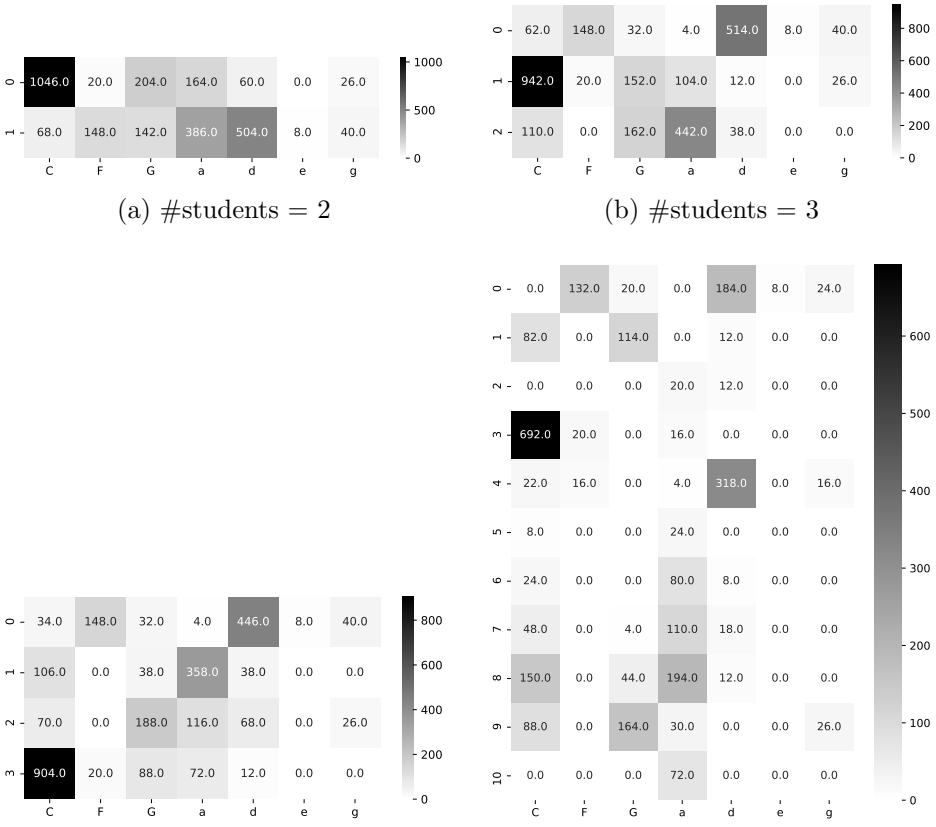


(b) Perplexities on the testing set.

Figure 5.3: Averaged perplexities by three trials with random seed of $\{0, 1, 2\}$. The number of students varies from 2 – 16.

are basically within closely related keys [75], which is consistent with the human annotation in **Table 5.3**. Obtained perplexity for development and testing sets are shown in **Figure 5.3**. We observed that a larger number of students contributed to a better score of perplexity. However, we also noticed in **Figure 5.3** that when the number of students was larger than 4 or 6, the improvement of perplexity became less significant. This finding is consistent with our intuition that the number of modes is not so many. However, we admit that the difficulty of finding an optimal number of students by perplexity solely.

5.4.2 Evaluation with a Human Analysis



(a) #students = 2 (b) #students = 3

(c) #students = 4 (d) #students = 11

Figure 5.4: Confusion matrices of clustering results, where the key that appeared more often in a phrase is chosen for pivot chords.

Next, we consult the human analysis and evaluate to what extent ob-

Table 5.4: Precision, Recall, and F1 scores of key detection. The human analysis [21] is used as a gold data.

#students	(i) choose the first			(ii) choose the last		
	P	R	F1	P	R	F1
2	56.11	**83.17**	67.01	55.97	**82.95**	66.84
3	67.47	79.90	**73.16**	67.83	79.83	**73.34**
4	66.76	73.37	69.91	66.76	73.44	69.94
5	65.06	63.57	64.30	65.48	63.42	64.44
6	65.98	56.04	60.60	67.40	56.39	61.41
7	64.99	50.71	56.97	66.26	51.07	57.68
8	68.75	59.66	63.88	69.03	59.87	64.13
9	64.63	50.36	56.61	64.77	50.57	56.80
10	68.04	46.95	55.56	68.61	47.30	56
11	**69.03**	53.84	60.49	**69.32**	53.34	60.29
12	64.42	38.78	48.41	65.34	39.99	49.61
13	65.48	40.48	50.03	65.84	41.12	50.62
14	67.61	37.64	48.36	67.54	38.85	49.33
15	67.76	37.29	48.10	67.33	36.58	47.40
16	66.76	37.71	48.20	66.83	37.86	48.33
#students	(iii) choose the more often			(iv) the most by phrase		
	P	R	F1	P	R	F1
2	55.04	**82.95**	66.18	55.42	**83.13**	66.51
3	67.40	80.11	**73.21**	68.07	82.53	**74.61**
4	67.33	74.29	70.64	70.48	77.71	73.92
5	65.34	64.56	64.95	69.28	68.07	68.67
6	65.98	57.10	61.22	66.87	60.24	63.38
7	64.99	51.49	57.46	66.87	51.81	58.38
8	69.60	60.3	64.62	**74.10**	63.25	68.25
9	64.63	51.14	57.10	65.66	51.81	57.92
10	68.75	47.66	56.29	71.69	47.59	57.20
11	**70.03**	54.47	61.28	**74.10**	56.63	64.19
12	64.42	39.84	49.23	66.27	41.57	51.09
13	65.91	41.55	50.97	67.47	42.77	52.35
14	67.97	38.71	49.32	71.69	40.36	51.65
15	68.18	37.71	48.56	71.69	36.75	48.59
16	67.83	38.71	49.29	72.89	40.36	51.95

tained clusters of modes are consistent with it. We create a confusion matrix $M_{confusion} = (d_{lr})$, where each element represents the counts of events classified to the cluster l by the model and key r in the human annotation. Then we calculate the Precision, Recall, and F1 scores as follows[11].

$$\text{Precision} = \frac{\sum_l \max_r(d_{lr})}{\sum_l \sum_r d_{lr}} \tag{5.3}$$

$$\text{Recall} = \frac{\sum_r \max_l(d_{lr})}{\sum_l \sum_r d_{lr}} \tag{5.4}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

For each obtained cluster of tonality l, we regard the corresponding gold tonality r as one where the maximum numbers of l are assigned: $\max_r(d_{lr})$. Then, Precision is calculated as (5.3). On the other hand, for each gold tonality r, corresponding l is assigned by $\max_l(d_{lr})$, and thus Recall is calculated as (5.4). Thus, Precision and Recall are in the trade-off relation. Therefore, we consult the F1 score (5.5), which is the harmonic mean of Precision and Recall.

The obtained scores are shown in **Table 5.4**. The human analysis often assigns two keys to *pivot chords*[12]. Therefore, we calculated the score with three settings for pivot chords: (i) choosing the first key, (ii) choosing the last key, and (iii) choosing the key that appeared more often in a phrase. We did not find a significant difference among the three settings in the obtained results. While modulations often occur within phrases, one of our model's drawbacks is that it can only detect tonality by phrase. Therefore, we also examined the score by (iv) selecting the most appeared key in a phrase from human annotations.

We found that the three-students model achieved the best F1 score in all settings. While a larger number of students contributed to the improvement of Precision, it degraded Recall as a trade-off. The confusion matrix for the three-students model (**Figure 5.4b**) showed that student 0 was mainly classified into **d** minor, student 1 to **C** major, and student 2 to **a** minor. While the human analysis [21] uses key labels of modern tonality, **d** minor would correspond to *dorian* mode. Discovered three modes (**C** major, **a** minor, *dorian*) are consistent with another analysis [22] where a considerable number of pieces are classified into *dorian* mode.

[11] The definition of Precision and Recall used here ((5.3) and (5.4)) is somewhat different from the common definition that assumes the number of classes in gold and prediction is the same. In this study, we use a known definition that is used for the different number of classes [87].

[12] A pivot chord is a chord that is shared by multiple keys.

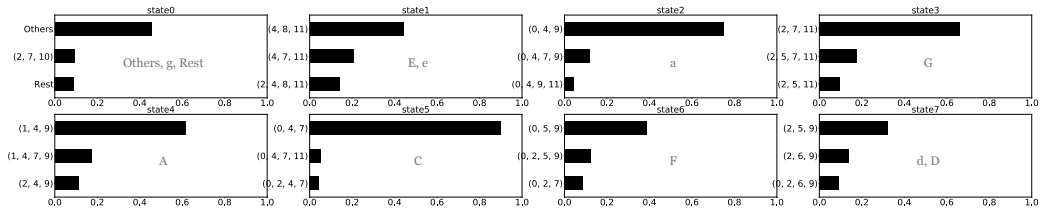
The second-best four-students model added **G** major cluster (**Figure 5.4c**). This finding is understandable since **G** major is the fourth most key in the human analysis (**Table 5.3**). Finally, in the 11-students model that scored the best Precision, we found that some students seemed to represent a mixture of keys. For example, while student 4 was a clear cluster of **d** minor (*dorian*), student 0 was a mixture of **d** minor and **F** major. In addition, there are several classes of a mixture of **a** minor and **C** major, where the proportions of the two keys are varied.

The findings that a larger number of students did not distinguish independent local keys and constructed mixtures of them would indicate the limitation of our model that detects tonality by phrase. Nevertheless, it is consistent with the previous findings of the seamless interchange between a minor key and its relative major key [73, 84].

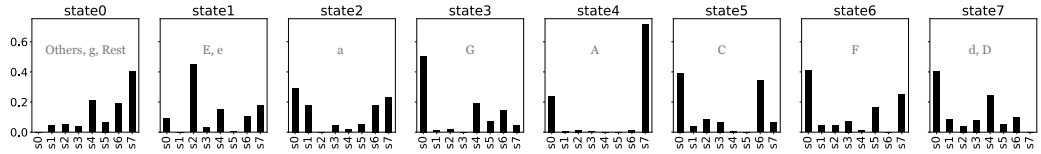
5.4.3 Discussion on Transition Probability

Finally, we show the obtained prototypes of chord transitions for the three discovered tonalities in **Figure 5.5b**, **Figure 5.5c**, and **Figure 5.5d**. Note that each hidden state represents a chord category, which is shown in **Figure 5.5a**. Although the clusters of tonalities were obtained in an unsupervised manner, the transition probabilities of which appropriately reflected the difference in the feature of chord transitions. For example, the dominant motion $\mathsf{G} \rightarrow \mathsf{C}$ and motion of subdominant or secondary dominant $\{\mathsf{d}, \mathsf{D}\} \rightarrow$ to dominant G are noticeable in student 1, which represents **C** major key. Unlike student 1, in student 2, which represents **a** minor key, G major chord (hidden state 3) has a little larger transition probability to a minor chord than C major chord.

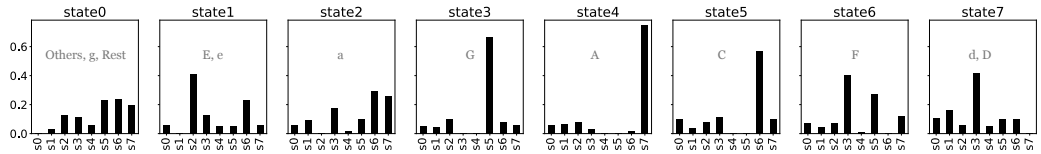
The transition probability from hidden state 7 (d minor or D major chords) shows the difference among the three tonalities. In student 0 (*dorian* mode), hidden state 7 (s_7) has a considerable transition probability to hidden state 0 (Others, or g as subdominant) and hidden state 4 (A major as dominant). On the other hand, in student 1 (**C** major key), s_7 has the largest transition probability to hidden state 3 (G major chord). Thus it seems to work as the subdominant or secondary dominant. Finally, in student 2 (**a** minor key), s_7 tends to proceed to s_1 (E major or e minor chord) as subdominant in **a** minor.



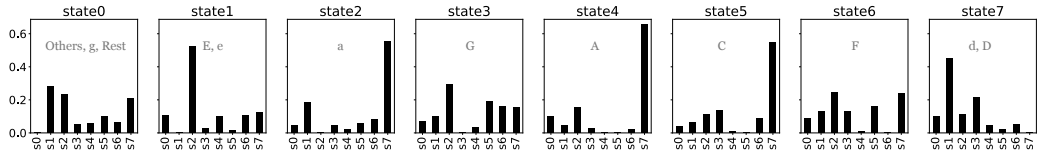
(a) Top three emission probabilities for each hidden state.



(b) The transition distribution of cluster 0 for the 3-students model.



(c) The transition distribution of cluster 1 for the 3-students model.



(d) The transition distribution of cluster 2 for the 3-students model.

Figure 5.5: Obtained emission and transition distributions for the 3-students model.

5.5 Chapter Summary

In this chapter, we extended the neural HSMM proposed in the previous chapter (Chapter 4) so as to be able to discover clusters of tonalities. In particular, we introduced a *teacher-student* architecture, where we employed the previous neural HSMM as the *teacher* model and prepared the *student* models that simplified the architecture of transition probability to multiple learnable matrices. The number of student matrices corresponded to the number of clusters of tonalities.

We classified tonalities by selecting the student matrix that achieved the highest similarity to the count of chord transitions given from the teacher model. The matrices were tuned by the gradient-based optimization with the loss of the negative log marginal likelihood by the student model. On the

other hand, other parameters were fixed and shared with the teacher model.

We expected the leaned transition matrices of students represented prototypes of tonalities. Since the teacher model was not designed to distinguish the difference of tonalities by itself, but students discovered it by the learning, students were not just simplified versions of the teacher model, in this sense.

We experimented with the proposed model with multiple settings of the number of students: 2–16 and evaluated the obtained clusters by consulting the human analysis. As a result, the 3-students model was the most consistent with the human analysis regarding the F1 score. By looking at the confusion matrix, we found that the obtained three clusters corresponded to *dorian*, **C** major, and **a** minor, respectively. In addition, the tuned transition matrices reflected the difference between tonalities, consistently with known tonic, dominant, and subdominant functions. Thus, our model found an appropriate cluster of tonalities without relying on human knowledge.

Chapter 6

Conclusion

6.1 Achievements

The objective of this study was to recognize chord categories, progression regularities, and tonality in a unified manner towards the statistical acquisition of inherent regularities in music. A fundamental difficulty here was that the three notions of harmony analysis, chords, chord functions, and tonality, are mutually dependent. Unlike preceding studies of unsupervised chord function identification that have disconnected this dependency in advance, we proposed models that analyzed these three notions considering their relationships.

To this end, we first proposed the neural HSMM that automatically learned chord categories and segmented surface notes to be classified into them in a situation where latent tonality could affect hidden state transition probabilities (Chapter 4). Here, we employed HSMM instead of HMM. Since HSMM equipped the notion of the duration of a chord, it enabled the model to segment notes without causing the useless increase of self-transitions. Thus, we could interpret the hidden state transition probabilities as learned chord progression regularities for the targeted pieces.

In the proposed model, neural networks were employed to calculate emission, hidden state transition, and duration distributions, where additional contexts, *e.g.,* preceding chord sequences, pitch-classes, and beat information, were embedded (Section 4.2). Such additional contexts were demonstrated to be effective by ablation studies to improve the model's generalization performance in terms of *perplexity*[1] (Section 4.5). The best eight-state model discovered clear chord clusters that mainly consisted of the chords on diatonic scales and commonly-used borrowed chords (Section 4.6.1). Further-

[1]Backgrounds for *perplexity* were described in Section 3.3.3.

more, transitions between chord categories were found to reflect the difference of tonalities with a tendency consistent with known chord functions by separately analyzing major, minor, and dorian pieces, which were manually classified.

Next, we extended the model to be able to classify tonalities in an unsupervised manner by employing the *teacher-student* architecture in Chapter 5. The teacher model was the same as the one proposed in Chapter 4, except for removing pitch histograms from additional contexts (Section 5.2.2). The student models, *i.e.,* multiple learnable matrices of hidden state transition probabilities, were tuned so as to become prototypes of tonalities (Section 5.2.3). We consulted the human analysis and evaluated to what extent obtained clusters of modes were consistent with it. As a result, the three-students model achieved the best F1 score, corresponding to *dorian*, **C** major, and **a** minor, respectively (Section 5.4.2). Furthermore, learned prototypes for tonalities (hidden state transition matrices) appropriately reflected the difference between the three tonalities, consistent with chord functions of tonic, dominant, and subdominant.

As summarized above, we proposed a model that recognizes chord categories, regularity of progression, and tonality in a unified manner using the neural HSMM and its extensions. Removing the prior annotation of key and chord segmentation presented a new difficulty, as it turned out that the baseline model could not find the appropriate chord clusters (Section 4.6). The proposed neural HSMM and additional contexts contributed to the solution.

6.2 Limitations and Future Directions

As the first step of giving up pre-annotated knowledge, we examined the proposed model on well-studied J. S. Bach's four-part chorales instead of proceeding to a broader range of pieces. The four-part chorales of J. S. Bach were expected to be described by the conventional theory for the most part but at the same time retained the feature of *church modes*. What was expected was appropriately demonstrated by the self-emergent chord functions and tonalities. Although we admit that the coverage of experimented pieces was limited, the achievements encourage us to apply our model to more diverse musical pieces in the future. However, this study remains several limitations before applying it to diverse musical styles and metrically more complex pieces. Our immediate future works are the following three issues, which may also be interrelated.

Unsupervised selection of an appropriate number of clusters

First, we have postponed selecting a unique optimal number of hid-

den states; we examined multiple numbers of them instead. Consistent with previous works [44, 82, 84], we found the difficulty in selecting the appropriate number of hidden states; a larger number of hidden states contributed to the improvement of *perplexity*. In addition, an appropriate number of hidden states may depend on the granularity at which we want to describe the knowledge. Nevertheless, studies based on Bayesian nonparametric methods suggest sophisticated approaches to select an optimal number of latent parameters [11, 58, 60]. Recent works have further studied the combination of the neural HMM and Bayesian methods [31, 63]. We plan to extend our model to automatically find an optimal number of latent states by utilizing these approaches.

Dynamic modulation detection

In Chapter 5, the proposed model of unsupervised clustering of tonality was limited to phrase-based clustering. However, the length of each phrase that was separated by the *fermata* position was usually two or three measures, which was not very large, and thus the matrix of counts of chord transitions was sparse. We found appropriate clusters of modes from such sparse count matrices. Therefore, we hope to extend it to a dynamic model. Another direction for dynamic modulation detection is employing multiple or hierarchical sequences of hidden states [27, 30]. In addition, combining both discriminative and generative models would be an interesting suggestion [9].

Extensions for more complex musical pieces

The surface structures of J. S. Bach's four-part chorale dataset that we used as the corpus for this study were relatively simple and coherent, regarding the rich diversity of musical pieces. Therefore, we would need further extensions to deal with such pieces that have more complex structures. We show an example of four-part harmonization (**Figure E.1**) and organ chorale (**Figure E.2**) in the Appendix. While these two are written based on the same hymn[2], the organ chorale is more elaborated.

We expect that the innate regularity in the granularity of chord functions and tonalities would not differ between the two pieces. On the other hand, we would need the chord segmentation to be more robust for the difference in surface structures. While we embedded surface information to some extent by the additional context of pitch-classes (Section 4.2), it would not be sufficient for more complicated pieces. A pos-

[2] *"Gelobet seist du, Jesu Christ"*

sible direction is extending the categorical emission distribution with a more flexible architecture like Variational Auto Encoder (VAE) [49] or Vector Quantized Variational Auto Encoder (VQ-VAE) [86]. The combination of HMM and VAE has been studied in unsupervised acoustic unit discovery, which was found to be more effective than the conventional HMM-GMM since observations would not follow Gaussian distribution in reality [24, 31].

By solving these immediate issues, we will be able to apply the unsupervised statistical learning of musical grammar to extensive musical pieces. We expect to find inherent regularities unique to each composer, especially those in the post romanticism era, which have not been studied well for their complex harmony.

Not only the finding of composer-dependent musical grammar would be important in musicology, but it also potentially contributes to computational systems that interact with humans in music understanding or creation. To this end, the proposed model conceptually presupposed notions of chords, chord function, and tonality as described in Chapter 1, unlike some recent studies aiming directly at music generations (compositions) [36, 43]. These notions have a long history and contributed to a broad range of musical activities. With such properly structured knowledge, humans can learn from examples and creatively utilize them. We hope to improve the proposed unsupervised statistical learning to provide such knowledge that reflects the unique characteristics of each composer.

Appendix A

Comparison with Conventional HSMM Trained by the EM Algorithm

As described in Section 4.3, the widely used expectation maximization (EM) algorithm possibly leads to bad local optima. We compare the conventional HSMM trained by EM with a neural HSMM of minimum architecture tuned by a gradient-based optimizer (RAdam [55]).

A.1 Baseline Neural HSMM

For comparison, we implement a minimum neural HSMM denoted as the *baseline* model; it represents distributions as simply learnable weight vectors or matrices with softmax output layers,

$$\begin{aligned}
\text{initial hidden state} &: \rho_i = P(z_0 = i) = \text{softmax}_i(\boldsymbol{\pi}) \\
\text{hidden state transition} &: a_{ij} = P(z_{t+1} = j | z_t = i) = \text{softmax}_j(\boldsymbol{v}_i^{trans.}) \\
\text{hidden state duration} &: p_{i\tau} = P(\tau | z_t = i) = \text{softmax}_\tau(\boldsymbol{v}_i^{dur.}) \\
\text{emission} &: b_{ik} = P(x_t = k | z_t = i) = \text{softmax}_k(\boldsymbol{v}_i^{emit})
\end{aligned}$$

where $\boldsymbol{\pi}$, $\boldsymbol{v}_i^{trans.}$, $\boldsymbol{v}_i^{dur.}$ and $\boldsymbol{v}_i^{emit}$ are (unnormalized) learnable vectors for initial state, state transition, state duration and emission distributions respectively. Given these distributions, the marginal likelihood for a sequence ($P(\mathbf{x}_{1:T})$) can be calculated by the forward algorithm, as shown in Section 3.3.2. The optimizer takes the negative log marginal likelihood ($-\log P(\mathbf{x}_{1:T})$) as loss and tunes the learnable vectors.

A.2 EM Algorithm for HSMM

In this section, we briefly describe the EM algorithm for HSMM, in particular, the Residential-time HMM [90, 91]. In addition to the forward algorithm (Section 3.3.2), we need to add the following processes.

A.2.1 Backward Algorithm

The backward variable is defined as follows [90, 91]:

$$\beta_t(j,\tau) \triangleq P(\mathbf{x}_{t+1:T}|z_{t:t+\tau-1]} = j)$$

where

$$\beta_t(j,\tau) = P(x_{t+1} = k|z_{t+1} = j)\beta_{t+1}(j,\tau-1) = b_{jk}\beta_{t+1}(j,\tau-1), \text{ for } \tau > 1.$$

$$\beta_t(j,1) = \sum_{i\backslash j} P(z_{t+1} = i|z_t = j)P(x_{t+1} = k|z_{t+1} = i)\Big(\sum_{\tau\geq 1} P(\tau|z_{t+1} = i)\beta_{t+1}(i,\tau)\Big)$$

$$= \sum_{i\backslash j} a_{ji}b_{ik}\Big(\sum_{\tau\geq 1} p_{i\tau}\beta_{t+1}(i,\tau)\Big)$$

Similar to the forward algorithm, we perform scaling to avoid underflow. The modified backward recursion is following.

$$\hat{\beta}_i(j,\tau) = \frac{\beta_t(j,\tau)}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}$$

$$C_{t+1}\hat{\beta}_t(j,\tau) = \frac{\beta_t(j,\tau)}{P(\mathbf{x}_{t+2:T}|\mathbf{x}_{1:t+1})} = b_{jk}\hat{\beta}_{t+1}(j,\tau-1)$$

$$C_{t+1}\hat{\beta}_t(j,1) = \frac{\beta_t(j,1)}{P(\mathbf{x}_{t+2:T}|\mathbf{x}_{1:t+1})} = \sum_{i\backslash j} a_{ji}b_{ik}\Big(\sum_{\tau\geq 1} p_{i\tau}\hat{\beta}_{t+1}(i,\tau)\Big)$$

where

$$C_{t+1} = P(x_{t+1}|\mathbf{x}_{1:t})$$

and the following equation is used.

$$P(\mathbf{x}_{t+2:T}|\mathbf{x}_{1:t+1}) = \frac{P(\mathbf{x}_{1:T})}{P(\mathbf{x}_{1:t+1})}$$

$$P(\mathbf{x}_{1:t+1}) = P(x_{t+1}|\mathbf{x}_{1:t})P(\mathbf{x}_{1:t}) = C_{t+1}P(\mathbf{x}_{1:t})$$

Note that C_t $(t = 1,\ldots,T)$ are calculated in the forward algorithm (see Section 3.3.2) then passed to the backward algorithm.

A.2.2 State Estimation

Similar to the EM algorithm for HMM, variable $\xi_t(i,j)$ and $\gamma_t(j)$ are introduced to estimate hidden states [10, 90, 91]. In addition, the variable $\eta_t(j,\tau)$ is introduced to estimate residential times. We describe them in the following.

$\xi_t(i,j)$:

$\xi_t(i,j)$ is defined as $\xi_t(i,j) \triangleq P(z_{t-1}=i, z_t=j, \mathbf{x}_{1:T}),\ i \neq j$ and is calculated as follows [91].

$$\xi_t(i,j) = \alpha_{t-1}(i,1)P(z_t=j|z_{t-1}=i)P(x_t=k|z_t=j)\Big(\sum_{\tau\geq 1} P(\tau|z_t=j)\beta_t(j,\tau)\Big)$$

$$= \alpha_{t-1}(i,1)a_{ij}b_{jk}\Big(\sum_{\tau\geq 1} p_{j\tau}\beta_t(j,\tau)\Big)$$

Here again, we apply a scaling as follows.

$$\begin{aligned}\hat{\xi}(i,j) \triangleq P(z_{t-1}=i, z_t=j|\mathbf{x}_{1:T}) &= \frac{\xi_t(i,j)}{P(\mathbf{x}_{1:T})} \\ &= \frac{1}{P(x_t|\mathbf{x}_{1:t-1})}\frac{\alpha_{t-1}(i,1)a_{ij}b_{jk}}{P(\mathbf{x}_{1:t-1})}\Big(\frac{\sum_{\tau\geq 1} p_{j\tau}\beta_t(j,\tau)}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})}\Big) \\ &= \frac{1}{C_t}\hat{\alpha}_{t-1}(i,1)a_{ij}b_{jk}\Big(\sum_{\tau\geq 1} p_{j\tau}\hat{\beta}_t(j,\tau)\Big)\end{aligned}$$

$\gamma_t(j)$:

$\gamma_t(j)$ is defined as $\gamma_t(j) \triangleq P(z_t=j, \mathbf{x}_{1:T})$. The recursion for $\gamma_t(j)$ is derived as follows, based on [91].

By using following equation,

$$\begin{aligned}P(z_t=j, z_{t+1}=j, \mathbf{x}_{1:T}) &= P(z_t=j, \mathbf{x}_{1:T}) - P(z_t=j, z_{t+1}\neq j, \mathbf{x}_{1:T}) \\ &= P(z_{t+1}=j, \mathbf{x}_{1:T}) - P(z_t\neq j, z_{t+1}=j, \mathbf{x}_{1:T})\end{aligned}$$

then,

$$\begin{aligned}\gamma_t(j) &\triangleq P(z_t = j, \mathbf{x}_{1:T}) \\ &= P(z_{t+1} = j, \mathbf{x}_{1:T}) + \Big(P(z_t = j, z_{t+1} \neq j, \mathbf{x}_{1:T}) - P(z_t \neq j, z_{t+1} = j, \mathbf{x}_{1:T})\Big) \\ &= \gamma_{t+1}(j) + \sum_{i \neq j} \Big(P(z_t = j, z_{t+1} = i, \mathbf{x}_{1:T}) - P(z_t = i, z_{t+1} = j, \mathbf{x}_{1:T})\Big) \\ &= \gamma_{t+1}(j) + \sum_{i \neq j} (\xi_{t+1}(j,i) - \xi_{t+1}(i,j))\end{aligned}$$

The initial condition for it is,

$$\gamma_T(j) = \sum_{\tau \geq 1} \alpha_T(j, \tau)$$

Since the above derivation still holds if we change $P(z_t = j, \mathbf{x}_{1:T})$ to $P(z_t = j | \mathbf{x}_{1:T})$, we have,

$$\hat{\gamma}_t(j) \triangleq P(z_t = j | \mathbf{x}_{1:T}) = \hat{\gamma}_{t+1}(j) + \sum_{i \neq j} \Big(\hat{\xi}_{t+1}(j,i) - \hat{\xi}_{t+1}(i,j)\Big)$$

$$\hat{\gamma}_T(j) = \sum_{\tau \geq 1} \hat{\alpha}_T(j, \tau)$$

$\eta_t(j, \tau)$:

$\eta_t(j, \tau)$ is defined as $\eta_t(j, \tau) \triangleq P(z_{t-1} \neq j, z_t = j, \tau, \mathbf{x}_{1:T})$ and is calculated as follows [91].

$$\begin{aligned}\eta_t(j, \tau) &\triangleq P(z_{t-1} \neq j, z_t = j, \tau, \mathbf{x}_{1:T}) \\ &= \Big(\sum_{i \neq j} \alpha_{t-1}(i, 1) P(z_t = j | z_{t-1} = i)\Big) P(x_t = k | z_t = j) P(\tau | z_t = j) \beta_t(j, \tau) \\ &= \Big(\sum_{i \neq j} \alpha_{t-1}(i, 1) a_{ij}\Big) b_{jk} p_{j\tau} \beta_t(j, \tau)\end{aligned}$$

Then, the scaling is applied as follows.

$$\begin{aligned}\hat{\eta}_t(j,\tau) &\triangleq \frac{\eta_t(j,\tau)}{P(\mathbf{x}_{1:T})} \\ &= \frac{1}{P(x_t|\mathbf{x}_{1:t-1})}\frac{\sum_{i\neq j}\alpha_{t-1}(i,1)a_{ij}}{P(\mathbf{x}_{1:t-1})}b_{jk}p_{j\tau}\frac{\beta_t(j,\tau)}{P(\mathbf{x}_{t+1:T}|\mathbf{x}_{1:t})} \\ &= \frac{1}{C_t}\Big(\sum_{i\neq j}\hat{\alpha}_t(i,1)a_{ij}\Big)b_{jk}p_{j\tau}\hat{\beta}_t(j,\tau)\end{aligned}$$

A.2.3 Parameter Re-estimation

Given $\xi_t(i,j)$, $\gamma_t(j)$ and $\eta_t(j,\tau)$, initial hidden state ρ_j, hidden state transition a_{ij}, hidden state duration $p_{j\tau}$, and emission b_{jk} parameters are re-estimated as follows. Note that $a_{ii} = 0$ for the Residential-time HMM (Section 3.3.2) [90, 91].

$$\begin{aligned}\rho_j &= \frac{\gamma_0(j)}{\sum_j \gamma_0(j)} \\ a_{ij} &= \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_j\sum_{t=1}^{T}\xi_t(i,j)},\ i\neq j \\ p_{j\tau} &= \frac{\sum_{t=1}^{T}\eta_t(j,\tau)}{\sum_{\tau\geq 1}\sum_{t=1}^{T}\eta_t(j,\tau)} \\ b_{jk} &= \frac{\sum_{t=1}^{T}\gamma_t(j)\delta(x_t,k)}{\sum_{t=1}^{T}\gamma_t(j)}\end{aligned}$$

where $\delta(x_t,k) = 1$ if $x_t = k$ otherwise 0.

A.3 Experiments

A.3.1 Setups

The same dataset and vocabulary as those described in Section 4.4 are used in this additional experiment. In addition, the same maximum epochs and early-stop settings in Section 4.4 are used. While the parameters in the baseline model are updated by mini-batch training (the mini-batch size is 8), the parameter re-estimation in the EM algorithm is conducted at every end of an epoch. Each value in the (unnormalized) learnable vectors for the baseline model is initialized by $\mathcal{N}(0,1)$. The parameter matrices for the conventional HSMM tuned by EM are initialized randomly using uniform distribution and normalized to meet the sum of probabilities to 1.

A.3.2 Results

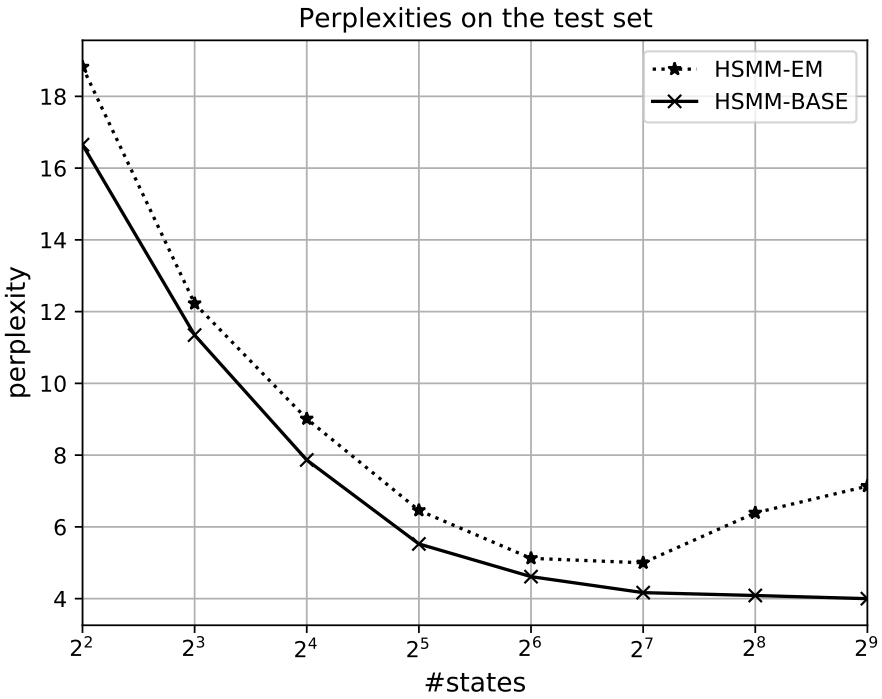


Figure A.1: Average perplexities of baseline neural HSMMs (solid) and conventional EM-trained HSMMs (dotted) for testing dataset

We compared the average perplexities[1] of baseline neural HSMMs and conventional HSMMs trained by EM in the number of hidden states {4, 8, 16, 32, 64, 128, 256, 512}. The scores of conventional HSMMs trained by EM showed rebound at numbers of hidden states larger than 128, which is a sign of overfitting. On the other hand, the baseline neural HSMMs seemed to converge into a certain minimum value. In addition, the baseline neural HSMMs outperformed EM models in terms of perplexity in all numbers of hidden states.

These results suggest that even the most straightforward neural HSMM baseline would have the merit of robustness compared to the EM-based conventional model. Note again that the neural HSMMs have shown further efficiency by elaborated architectures with additional contexts, as reported in Section 4.5.

[1]Here, we used the revised version of average perplexity (a) described in Chapter B.

Appendix B

Revised Average Perplexity

In Section 4.5, we have reported the average perplexity calculated as follows,

$$\text{Average perplexity (originally reported)} = \frac{\sum_{n=1}^{N_d} \exp\Big(-\frac{1}{T^{(n)}} \log P(\mathbf{x}_{1:T^{(n)}})\Big)}{N_d}$$

$$= \frac{\sum_{n=1}^{N_d} \exp\Big(-\frac{1}{T^{(n)}} \sum_{t=1}^{T^{(n)}} \log P(x_t|x_1,\ldots,x_{t-1})\Big)}{N_d}$$

where N_d is the number of testing sequences and $T^{(n)}$ is the sequence length of a testing sequence $\mathbf{x}_{1:T^{(n)}}$; this value is a simple average of the perplexity of each sequence.

However, considering the perplexity can be interpreted as an approximated cross-entropy between the actual (but unknown) probability and an estimated probability as mentioned in Section 3.3.3, we report revised perplexities calculated as follows[1][2].

$$\text{Average perplexity (revised(a))} = \exp\Big(-\frac{\sum_{n=1}^{N_d} \log P(\mathbf{x}_{1:T^{(n)}})}{\sum_{n=1}^{N_d} T^{(n)}}\Big)$$

$$= \exp\Big(-\frac{\sum_{n=1}^{N_d} \sum_{t=1}^{T^{(n)}} \log P(x_t|x_1,\ldots,x_{t-1})}{\sum_{n=1}^{N_d} T^{(n)}}\Big)$$

Furthermore, if the n-gram model $P(x_t|x_1,\ldots,x_{t-1})$ is different for each

[1] In addition, we also modified the sequence length $T^{(n)}$ not to count the artificial start symbol.

[2] Note that we further take a simple average with three random seeds in both scores of original and revised average perplexities.

sequence, the following definition should be applied [57].

$$\text{Average perplexity (revised(b))} = \exp\Big(-\frac{1}{N_d}\sum_{n=1}^{N_d}\frac{1}{T^{(n)}}\log P(\mathbf{x}_{1:T^{(n)}})\Big)$$

$$= \exp\Big(-\frac{1}{N_d}\sum_{n=1}^{N_d}\frac{1}{T^{(n)}}\sum_{t=1}^{T^{(n)}}\log P(x_t|x_1,\ldots,x_{t-1})\Big)$$

Although the original average perplexity (**Table B.1**) and revised ones (**Table B.2**, **Table B.3**) show similar results, the revised calculations would be more appropriate considering the background of the perplexity. In our case, both revised perplexities (a) and (b) could be used; the additional context of the histogram (4.2) may implicitly change the n-gram probability for sequence, but the other contexts do not break the n-gram assumption[3].

Table B.1: Average perplexity (originally reported as **Table 4.3**)

#hidden states	4	8	16	32
BASE	16.47	11.42	7.96	5.64
NHSMM	**14.43**	**9.36**	**6.14**	**4.74**
−HISTO	15.18	9.78	6.51	4.92
−LSTM	14.94	9.67	6.47	5.07
−BEAT	14.85	9.65	6.29	4.77
−PITCH	16.88	10.64	7.23	5.38

Table B.2: Revised average perplexity (a)

#hidden states	4	8	16	32
BASE	16.65	11.35	7.86	5.52
NHSMM	**14.25**	**9.14**	**5.97**	**4.62**
−HISTO	14.91	9.45	6.33	4.78
−LSTM	14.65	9.49	6.37	4.97
−BEAT	14.49	9.42	6.16	4.65
−PITCH	16.46	10.21	6.89	5.11

[3]In addition, the additional contexts are not random variables like the input features in Input-Output HMM [7].

Table B.3: Revised average perplexity (b)

#hidden states	4	8	16	32
BASE	16.64	11.37	7.88	5.53
NHSMM	**14.21**	**9.12**	**5.95**	**4.61**
−HISTO	14.89	9.44	6.31	4.78
−LSTM	14.63	9.47	6.33	4.96
−BEAT	14.47	9.41	6.13	4.63
−PITCH	16.46	10.21	6.89	5.12

Appendix C

Notations and Model Settings

C.1 Notations

Table C.1: Notations in the neural HSMM.

N	: number of sequences in a mini-batch
$n \in N$	: a sequence index in a mini-batch
S	: number of hidden states
V	: vocabulary size
T	: maximum time step of a sequence
$t \in T$	: time step
z_t	: hidden state at t
$i, j \in S$	: state indices
D	: maximum hidden state duration
$\tau \in D$	: duration index
x_t	: index of observed pitch-class content at t
$k \in V$	: observed symbol index
$\mathbf{x}_{1:T}$	: an observed sequence
$p_{i\tau}$	: duration probability
ρ_i	: initial hidden state probability
$\boldsymbol{\pi}$	: unnormalized initial hidden state probability
a_{ij}	: hidden state transition probability
b_{ik}	: emission probability
$\boldsymbol{s}_i$	: hidden state embedding
$\boldsymbol{v}_k^{pitch}$	: binary pitch-class vector
$\boldsymbol{v}^{histo}$	: observed pitch-class histogram
$\boldsymbol{r}^{histo}$	: pitch-class histogram encoding
$\boldsymbol{o}_k$	: observation encoding
l_k	: emission bias

C.2 Model Settings

Table C.2: The size of layers and related equations.

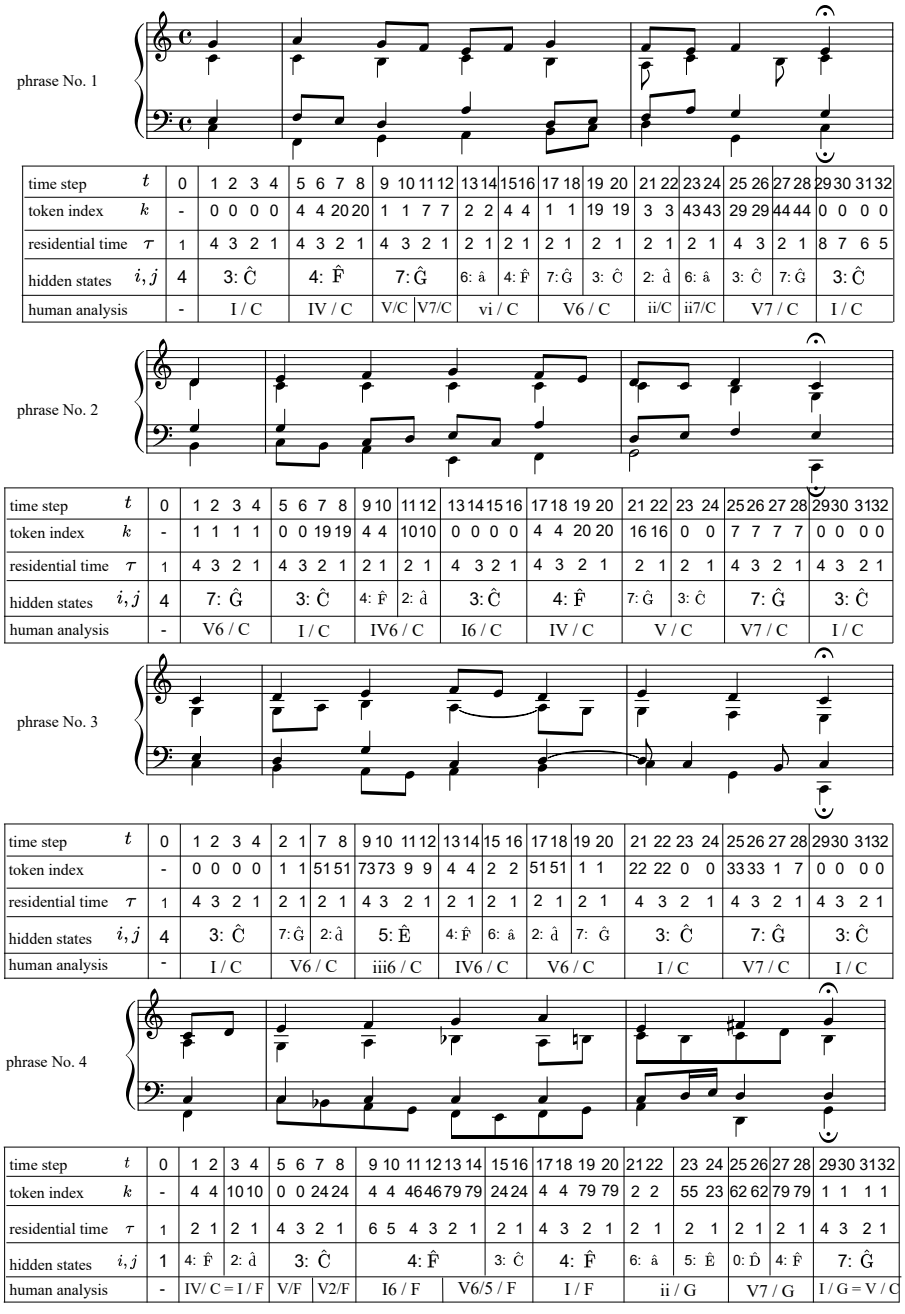
layer	size	eq.
hidden layer size of MLP_3 for transition and duration probability	16	(4.1), (4.6)
dimension of hidden state embedding vector ($\boldsymbol{s}_i$)	16	(4.1), (4.6), (4.8)
hidden layer sizes of MLP_2 for histogram embedding	8	(4.2)
hidden and layer sizes of MLP_2 for observation embedding	16	(4.3)
dimension of observation embedding vector ($\boldsymbol{o}_k$)	16	(4.3), (4.8)
dimension of LSTM hidden vector $\boldsymbol{h}_t$	16	(4.4)
hidden layer size of MLP_2 for beat encoder	8	(4.7)
dimension of beat embedding vector $\boldsymbol{r}_t^{beat}$	8	(4.7)

C.3 Implementation

The code is available at:
https://github.com/yui-u/emerge-chord-function

Appendix D

Analysis on BWV267 by the neural HSMM



time step t	0	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21 22	23 24	25 26 27 28	29 30 31 32
token index k	-	0 0 0 0	4 4 20 20	1 1 7 7	2 2 4 4	1 1 19 19	3 3	43 43	29 29 44 44	0 0 0 0
residential time τ	1	4 3 2 1	4 3 2 1	4 3 2 1	2 1 2 1	2 1 2 1	2 1	2 1	4 3 2 1	8 7 6 5
hidden states i,j	4	3: Ĉ	4: F̂	7: Ĝ	6: â 4: F̂	7: Ĝ 3: Ĉ	2: d̂	6: â	3: Ĉ 7: Ĝ	3: Ĉ
human analysis	-	I / C	IV / C	V/C	V7/C vi / C	V6 / C	ii/C	ii7/C	V7 / C	I / C

time step t	0	1 2 3 4	5 6 7 8	9 10	11 12	13 14 15 16	17 18 19 20	21 22	23 24	25 26 27 28	29 30 31 32
token index k	-	1 1 1 1	0 0 19 19	4 4	10 10	0 0 0 0	4 4 20 20	16 16	0 0	7 7 7 7	0 0 0 0
residential time τ	1	4 3 2 1	4 3 2 1	2 1	2 1	4 3 2 1	4 3 2 1	2 1	2 1	4 3 2 1	4 3 2 1
hidden states i,j	4	7: Ĝ	3: Ĉ	4: F̂	2: d̂	3: Ĉ	4: F̂	7: Ĝ	3: Ĉ	7: Ĝ	3: Ĉ
human analysis	-	V6 / C	I / C	IV6 / C		I6 / C	IV / C	V / C		V7 / C	I / C

time step t	0	1 2 3 4	2 1	7 8	9 10	11 12	13 14	15 16	17 18	19 20	21 22 23 24	25 26 27 28	29 30 31 32
token index	-	0 0 0 0	1 1	51 51	73 73	9 9	4 4	2 2	51 51	1 1	22 22 0 0	33 33 1 7	0 0 0 0
residential time τ	1	4 3 2 1	2 1	2 1	4 3 2 1		2 1	2 1	2 1	2 1	4 3 2 1	4 3 2 1	4 3 2 1
hidden states i,j	4	3: Ĉ	7: Ĝ	2: d̂	5: Ê		4: F̂	6: â	2: d̂	7: Ĝ	3: Ĉ	7: Ĝ	3: Ĉ
human analysis	-	I / C	V6 / C		iii6 / C		IV6 / C		V6 / C		I / C	V7 / C	I / C

time step t	0	1 2	3 4	5 6 7 8	9 10 11 12 13 14	15 16	17 18 19 20	21 22	23 24	25 26	27 28	29 30 31 32
token index k	-	4 4	10 10	0 0 24 24	4 4 46 46 79 79	24 24	4 4 79 79	2 2	55 23	62 62	79 79	1 1 1 1
residential time τ	1	2 1	2 1	4 3 2 1	6 5 4 3 2 1	2 1	4 3 2 1	2 1	2 1	2 1	2 1	4 3 2 1
hidden states i,j	1	4: F̂	2: d̂	3: Ĉ	4: F̂	3: Ĉ	4: F̂	6: â	5: Ê	0: D̂	4: F̂	7: Ĝ
human analysis	-	IV/ C = I / F		V/F V2/F	I6 / F V6/5 / F		I / F	ii / G		V7 / G		I / G = V / C

Figure D.1: Chord classification by the neural HSMM on BWV267 (phrase No.1–4)

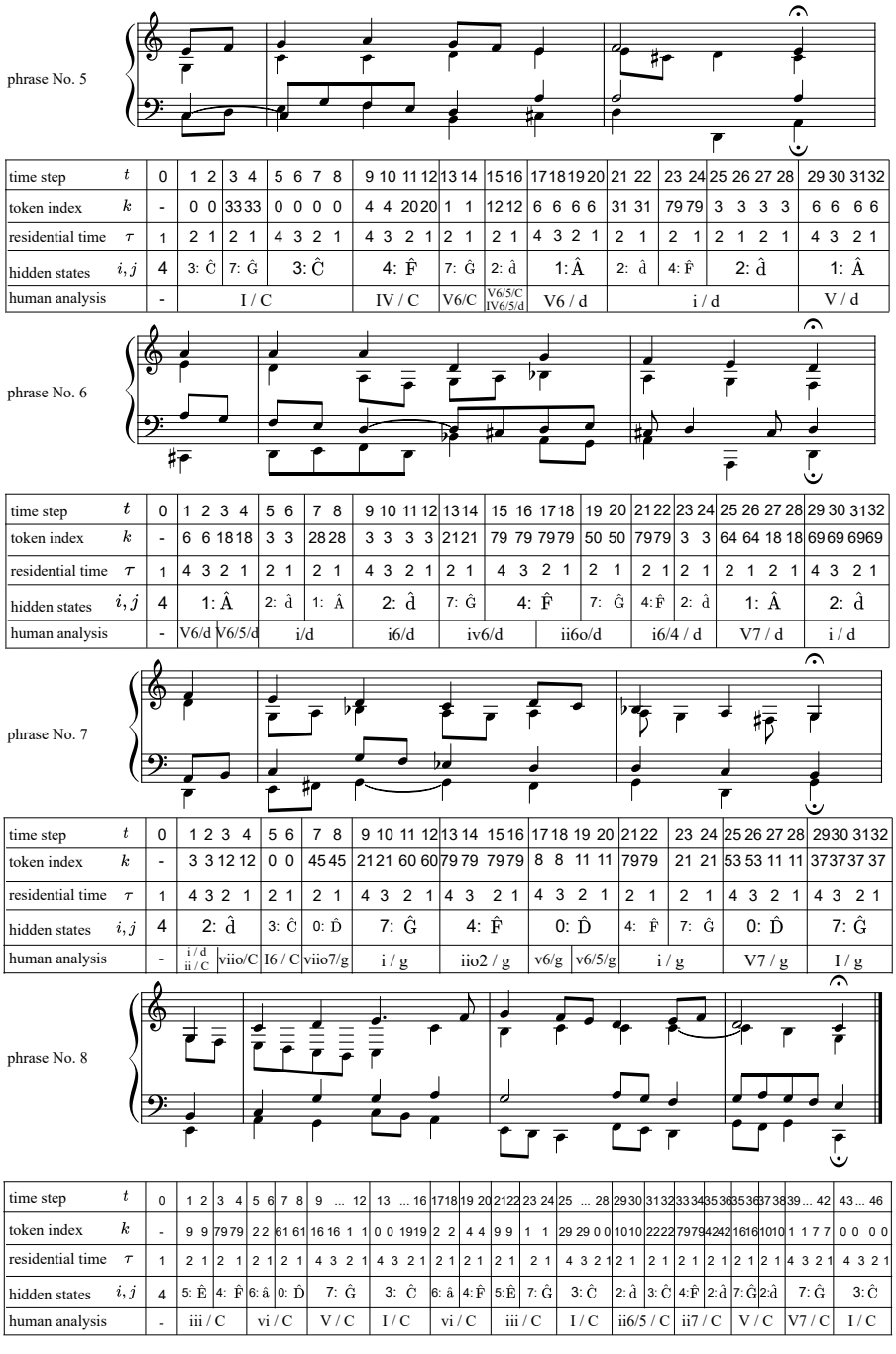


Figure D.2: Chord classification by the neural HSMM on BWV267 (phrase No.5–8)

Appendix E

An Example of Chorale and Instrumental Music

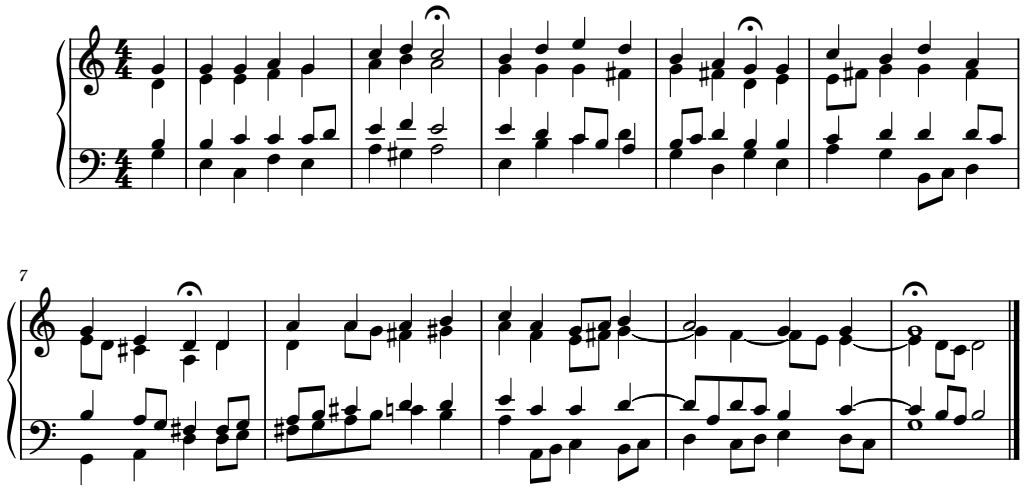


Figure E.1: Four-part chorale BWV64.2 (Riemenschneider No.160) by J. S. Bach

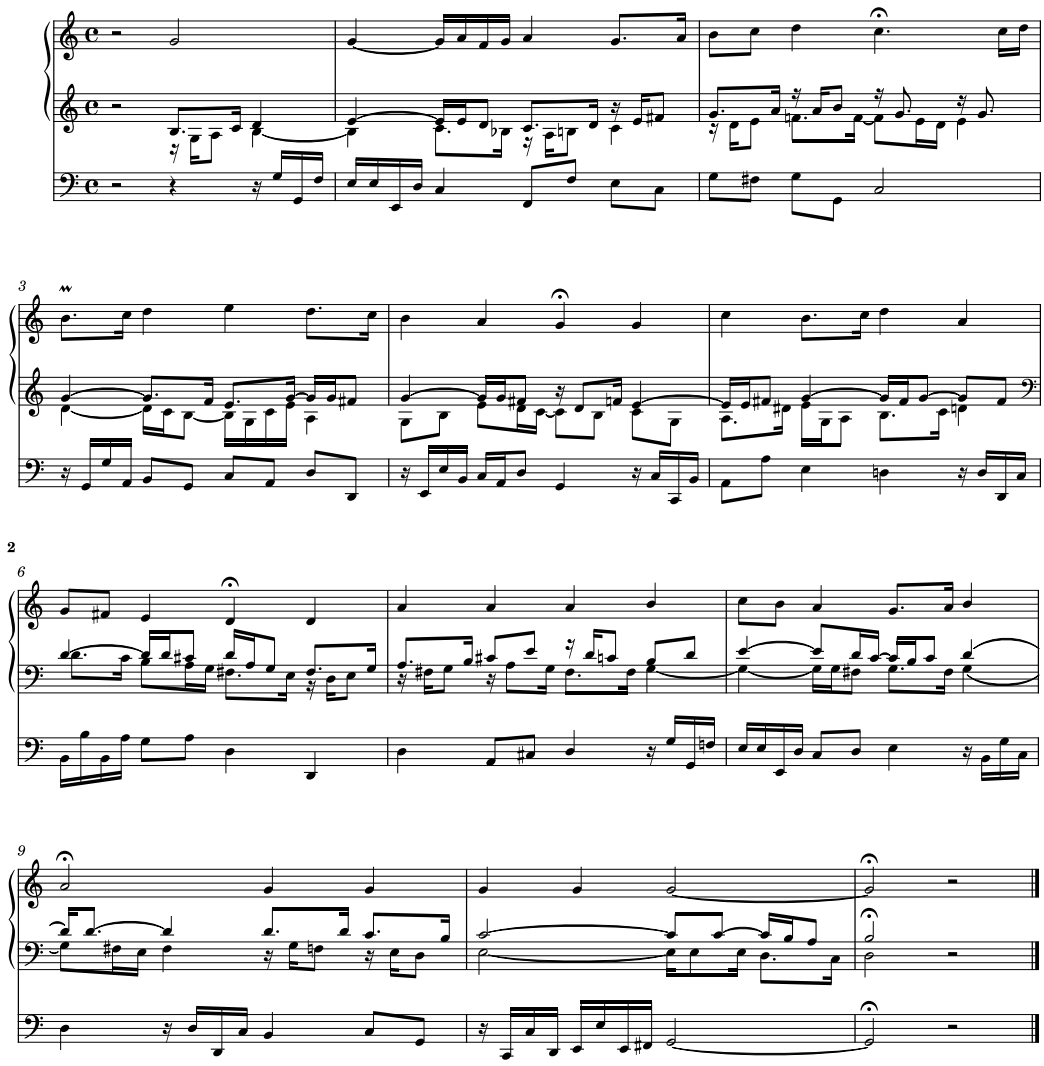


Figure E.2: Organ chorale BWV604 by J. S. Bach

Glossary

authentic cadence is a cadence of dominant to tonic . 7

borrowed chord is a chord that cannot be described by a chord degree in the current key but by another key . 7

chord is a combination of pitches that concurrently sound . 80

chord degree is the degree between the root note of a chord and that of the tonic chord . 6

chord function is a category or role of a chord that is defined based on the regularity of chord progressions . 5

church mode or a Gregorian mode is a modal scale used in medieval music that is often categorized by 8 – 12 modes . 13, 43, 52, 61

dorian is one of the church modes, the tonic of which is `d` . 13, 27, 36, 39–41, 44, 45, 56, 57, 59, 61

modulation is a change of tonality . 7, 46

Neapolitan II is a transformed chord, the root note of which is ♭**II** . 7

root note is the bottom note for a chord arranged as tertian harmony . 6, 80

secondary dominant is a chord that has a dominant function to the following chord . 7, 40, 57

tertian harmony is the basis of chords that are constructed with thirds . 7

Acronyms

CNN Convolutional Neural Network. 25

D dominant. 5–8, 11, 12, 40, 41, 59, 61

DNN Deep Neural Network. 25

EM expectation maximization. 16, 24, 35, 64–66, 68, 69

FNN feedforward neural network. 21

GMM Gaussian Mixture Model. 25

HMM hidden Markov model. v, 3, 4, 12–16, 18, 24–27, 35, 43, 46, 60, 62, 63, 66

HSMM hidden semi-Markov model. 3, 4, 14, 15, 18, 19, 27, 28, 30, 31, 33–36, 38–40, 43, 45, 46, 48, 50, 58, 60, 61, 64, 65, 68, 69

KL divergence Kullback-Leibler divergence. 46, 47

LSTM Long-Short Term Memory. 22, 23, 25, 32, 37, 50

MLE maximum likelihood estimation. 24

MLP multi layer perceptron. 21, 37

NLP Natural Language Processing. 1, 24

PCFG probabilistic context-free grammar. 12

POS part-of-speech. 1, 15, 24

RNN Recurrent Neural Network. 22, 23, 32

S subdominant. 5–7, 11, 12, 40, 41, 57, 59, 61

T tonic. 5–8, 11, 12, 40, 41, 59, 61

tanh hyperbolic tangent function. 21–23

VAE Variational Auto Encoder. 63

VQ-VAE Vector Quantized Variational Auto Encoder. 63

Bibliography

[1] B. J. Aarden. *Dynamic melodic expectancy*. The Ohio State University, 2003.

[2] J. Albrecht and D. Shanahan. The use of large corpora to train a new type of key-finding algorithm: An improved treatment of the minor mode. *Music Perception: An Interdisciplinary Journal*, 31(1):59–67, 2013.

[3] E. Aldwell, C. Schachter, and A. Cadwallader. *Harmony and voice leading*. Cengage Learning, Inc, 2018.

[4] T. Anders and E. R. Miranda. A computational model that generalises schoenberg's guidelines for favourable chord progressions. In *6th Sound and Music Computing Conference.*, 2009.

[5] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28, 2012.

[6] H. Bellmann. About the determination of key of a musical excerpt. In R. Kronland-Martinet, T. Voinier, and S. Ystad, editors, *Computer Music Modeling and Retrieval*, pages 76–91, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Y. Bengio and P. Frasconi. Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[8] T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California, June 2010. Association for Computational Linguistics.

[9] G. Bideault, L. Mioulet, C. Chatelain, and T. Paquet. A hybrid crf/hmm approach for handwriting recognition. In *International Conference Image Analysis and Recognition*, pages 403–410. Springer, 2014.

[10] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[11] Y. Bisk and J. Hockenmaier. An HDP Model for Inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88, 03 2013.

[12] P. Blunsom and T. Cohn. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, 2011.

[13] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1881–1888, Madison, WI, USA, 2012. Omnipress.

[14] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

[15] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[16] J.-T. Chien and Y.-C. Ku. Bayesian recurrent neural network for language modeling. *IEEE transactions on neural networks and learning systems*, 27(2):361–374, 2015.

[17] C. Christodoulopoulos, S. Goldwater, and M. Steedman. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, 2010.

[18] C. Christodoulopoulos, S. Goldwater, and M. Steedman. A bayesian mixture model for pos induction using multiple features. In *Proceedings of the 2011 conference on empirical methods in Natural Language Processing*, pages 638–647, 2011.

[19] A. Clark. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, Apr. 2003. Association for Computational Linguistics.

[20] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[21] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, 2010.

[22] L. Dahn. So how many bach four–part chorales are there?, 2018. http://www.bach-chorales.com/HowManyChorales.htm.

[23] D. de La Motte and V. Primožič. *Harmonielehre*. Bärenreiter, 1980.

[24] J. Ebbers, J. Heymann, L. Drude, T. Glarner, R. Haeb-Umbach, and B. Raj. Hidden markov model variational autoencoder for acoustic unit discovery. In *InterSpeech*, pages 488–492, 2017.

[25] S. R. Eddy. What is a hidden markov model? *Nature biotechnology*, 22(10):1315–1316, 2004.

[26] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé. Estimating keys and modulations in musical pieces. In *18th Sound and Music Computing Conference*, 2020.

[27] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.

[28] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[29] M. Gales and S. Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, Jan. 2007.

[30] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine learning*, 29(2):245–273, 1997.

[31] T. Glarner, P. Hanebrink, J. Ebbers, and R. Haeb-Umbach. Full bayesian hidden markov model variational autoencoder for acoustic unit discovery. In *Interspeech*, pages 2688–2692, 2018.

[32] S. Goldwater and T. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 744–751, 2007.

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[34] M. Granroth-Wilding and M. Steedman. Statistical parsing for harmonic analysis of jazz chord sequences. In *International Computer Music Conference*, pages 478–485, 2012.

[35] R. Groves. Automatic harmonization using a hidden semi-markov model. In *AIIDE Workshop*, pages 48–54, 2013.

[36] G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371. PMLR, 2017.

[37] D. Harasim, F. C. Moss, M. Ramirez, and M. Rohrmeier. Exploring the foundations of tonality: statistical cognitive modeling of modes in the history of Western classical music. *Humanities and Social Sciences Communications*, 8(1):5, 2021.

[38] M. R. Hassan and B. Nath. Stock market forecasting using hidden markov model: a new approach. In *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, pages 192–196. IEEE, 2005.

[39] H. Hermansky, D. P. W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 3, pages 1635–1638, 2000.

[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

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

[42] D. J. Hu and L. K. Saul. A probabilistic topic model for unsupervised learning of musical key-profiles. In *10th International Society for Music Information Retrieval Conference*, pages 441–446, 2009.

[43] C.-Z. 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.

[44] N. Jacoby, N. Tishby, and D. Tymoczko. An information theoretic approach to chord categorization and functional harmony. *Journal of New Music Research*, 44(3):219–244, 2015.

[45] M. Johnson. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[46] B. Juang and L. R. Rabiner. A probabilistic distance measure for hidden markov models. *AT&T technical journal*, 64(2):391–408, 1985.

[47] D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[48] Y. Kim, S. Wiseman, and A. M. Rush. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*, 2018.

[49] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk. Annotator subjectivity in harmony annotations ofpopular music. *Journal of New Music Research*, 48(3):232–252, 2019.

[51] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Oxford University Press, 2001.

[52] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[53] F.-F. Kuo and M.-K. Shan. A personalized music filtering system based on melody style classification. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 649–652. IEEE, 2002.

[54] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, 1983.

[55] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[56] N. N. López, L. Feisthauer, F. Levé, and I. Fujinaga. On local keys, modulations, and tonicizations. In *7th Digital Libraries for Musicology*, 2020.

[57] Y. Miao, E. Grefenstette, and P. Blunsom. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2410–2419. JMLR.org, 2017.

[58] D. Mochihashi, T. Yamada, and N. Ueda. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, Aug. 2009. Association for Computational Linguistics.

[59] J. Mulholland and T. Hojnacki. *The Berklee Book of Jazz Harmony*. Berklee Press, 2013.

[60] M. Nakano, J. L. Roux, H. Kameoka, T. Nakamura, N. Ono, and S. Sagayama. Bayesian nonparametric spectrogram modeling based on infinite factorial infinite hidden markov model. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 325–328, 2011.

[61] M. Navarro, M. Caetano, G. Bernardes, L. Nunes de Castro, and J. M. Corchado. Automatic generation of chord progressions with an artificial immune system. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 175–186, 2015.

[62] J.-F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR)*, 2005.

[63] A. Pakman, Y. Wang, C. Mitelut, J. Lee, and L. Paninski. Neural clustering processes. In *International Conference on Machine Learning*, pages 7455–7465. PMLR, 2020.

[64] H. Papadopoulos and G. Peeters. Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *2007 International Workshop on Content-Based Multimedia Indexing*, pages 53–60. IEEE, 2007.

[65] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[67] W. Piston and M. DeVoto. *Harmony*. W. W, Norton & Company, 1978.

[68] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[69] J.-P. Rameau. *Treatise on Harmony*. Dover Publications, 1971.

[70] H. Riemann. *Harmony Simplified: Or the Theory of the Tonal Functions of Chords*. Augener, 1896.

[71] A. Riemenschneider, editor. *371 harmonized chorales and 69 chorale melodies with figured bass by Johann Sebastian Bach*. G. Schirmer, Inc., 1986.

[72] M. Rohrmeier. Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[73] M. Rohrmeier and I. Cross. Statistical properties of tonal harmony in bach's chorales. In *10th International Conference on Music Perception and Cognition*, pages 619–627, 2008.

[74] H. Schenker. *Harmony*. University of Chicago Press, 1957.

[75] A. Schoenberg. *Structural Functions of Harmony (revised edition)*. W. W. Norton & Company, 1969.

[76] H. Tanie, T. Inamura, and Y. Nakamura. Construction of proto-symbol space in which stored motion pattern by continuous hmms adopted topological structure. In *The 17th Annual Conference of the Japanese Society for Artificial Intelligence*, pages 215–215, 2003.

[77] D. Temperley. The tonal properties of pitch-class sets: Tonal implication, tonal ambiguity, and tonalness. *Computing in Musicology*, 15, 2007.

[78] D. Temperley. The tonal properties of pitch-class sets: Tonal implication, tonal ambiguity, and tonalness. *Computing in Musicology*, 15, 2007.

[79] D. Temperley and E. W. Marvin. Pitch-class distribution and the identification of key. *Music Perception*, 25(3):193–212, 2008.

[80] K. Tran, Y. Bisk, A. Vaswani, D. Marcu, and K. Knight. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, 2016.

[81] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Function- and rhythm- aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences. In *Proceedings of 18th International Society for Music Information Retrieval Conference*, pages 502–508, 2017.

[82] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii. Generative statistical models with self-emergent grammar of chord sequences. *Journal of New Music Research*, 47(3):226–248, 2018.

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

[84] Y. Uehara, E. Nakamura, and S. Tojo. Chord function identification with modulation detection based on hmm. In R. Kronland-Martinet, S. Ystad, and M. Aramaki, editors, *Perception, Representations, Image, Sound, Music*, pages 166–178, Cham, 2021. Springer.

[85] Y. Uehara and S. Tojo. The simulated emergence of chord function. In J. Romero, T. Martins, and N. Rodríguez-Fernández, editors, *Artificial Intelligence in Music, Sound, Art and Design*, pages 264–280, Cham, 2021. Springer.

[86] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017.

[87] Y. Wang, H. C. Leung, S. Yiu, and F. Y. Chin. MetaCluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample. *Bioinformatics*, 28(18):i356–i362, 09 2012.

[88] C. W. White and I. Quinn. Chord Context and Harmonic Function in Tonal Music. *Music Theory Spectrum*, 40(2):314–335O, 11 2018.

[89] D. Yu, L. Deng, and G. E. Dahl. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *NIPS 2010 workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[90] S.-Z. Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010.

[91] S.-Z. Yu and H. Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden markov model. *IEEE signal processing letters*, 10(1):11–14, 2003.

Publications

Journal papers

[1] Y. Uehara and S. Tojo. Chord Function Recognition as Latent State Transition. *SN Computer Science*, (Accepted).

[2] Y. Uehara, S. Tojo, and R. Uehara. Unsupervised Discovery of Tonality in Bach's Chorales. *Journal of Intelligence Informatics and Smart Technology*, volume 8, October 2022, (In Press).

International conferences

[3] Y. Uehara, S. Tojo, and R. Uehara. Unsupervised Discovery of Tonality in Bach's Chorales. In *Proceedings of the Sixteenth International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2021)*, 2021.

[4] Y. Uehara and S. Tojo. The Simulated Emergence of Chord Function. In: J. Romero, T. Martins, N. Rodríguez-Fernández, editors, *Artificial Intelligence in Music, Sound, Art and Design. EvoMUSART 2021, Lecture Notes in Computer Science*, vol 12693, pages 264-280, Springer, Cham, 2021.

[5] H. Yamamoto, Y. Uehara, and S. Tojo. Jazz Harmony Analysis with ϵ-Transition and Cadential Shortcut. In *Proceedings of the 17th Sound and Music Computing Conference (SMC2020)*, pages 316-322, 2020.

[6] Y. Ogura, H. Ohmura, Y. Uehara, S. Tojo, and K. Katsurada. Expectation-based Parsing for Jazz Chord Sequences. In *Proceedings of the 17th Sound and Music Computing Conference (SMC2020)*, pages 350-356, 2020.

[7] Y. Uehara, E. Nakamura, and S. Tojo. Chord Function Identification with Modulation Detection Based on HMM. In: R. Kronland-Martinet, S. Ystad, and M. Aramaki, editors, *Perception, Representations, Image, Sound, Music. CMMR 2019, Lecture Notes in Computer Science*, vol 12631, pages 166-178, Springer, Cham, 2021.