Abstract of thesis entitled

Large Vocabulary Automatic Chord Estimation from Audio Using Deep Learning Approaches

submitted by

Junqi Deng

for the degree of Doctor of Philosophy
at The University of Hong Kong
August 31st, 2016

Being well aware of the chord annotation subjectivity issue, this thesis attests the necessity of large vocabulary with a joint argument of machine musicianship and the Turing test. Built upon this premise, it proposes two deep learning based system frameworks that lead to potential practical solutions to large vocabulary automatic chord estimation.

The first framework separates chord segmentation and classification into two tasks, which is unlike all previous approaches that combine them in one single pass. Several deep learning models are implemented and tested. Under the large vocabulary evaluation, the recurrent neural network model shows great potential in balanced performances across different chords. This framework has shown its advantages over large vocabulary evaluation in the automatic chord estimation task of music information retrieval evaluation exchange 2016.

The second framework incorporates a skewed class distribution sensitive approach. It employs an "even chance" scheme to boost the uncommon chords' exposure when training a recurrent neural network sequence decoder. The main drawback of this approach is the low segmentation quality. Nevertheless, it demonstrates the even chance training scheme to be effective for the large vocabulary automatic chord estimation.

Finally, a preliminary study has been conducted for automatic jazz chord estimation. Upon this study, a chord-scale estimation system is built and some semi-automatic or fully automatic jazz improvisation demos are created.

(Total words: 217)

Junqi Deng

Large Vocabulary Automatic Chord Estimation from Audio Using Deep Learning Approaches

by

Junqi Deng

B.Eng., M.Eng.

A thesis submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Department of Electrical and Electronic Engineering)

at

The University of Hong Kong

August 31st, 2016

Declaration

I declare that this thesis represents my own work, except where due acknowledgment is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree diploma or other qualifications.

Signed _____
Junqi Deng

To my loved ones

Acknowledgments

Firstly, I would like to thank my supervisor, Professor Ricky Yu-Kwong Kwok, for his kindly and thoughtful support during my PhD study. The works presented in this thesis, to some degree, are out of my pure personal interest in searching for a crossing point of music and computational technology. They can not even be ever started, if I was not given enough trust and encouragement. Throughout these 4 years time, Professor Kwok and I have a lot of constructive and meaningful conversations about the dual mission of career and life, which I believe are beneficial in the rest of my life.

I also thank Professor Francis Chi-moon Lau, for his appreciation of my early works. In the first two years, we communicate a lot in terms of music expression interfaces. I am deeply impressed by his enthusiasm and sophisticated knowledge in both computer and music. His comments make a lot of difference in this thesis.

Thank Doctor Xiao Hu, one of the most influential researchers in music information retrieval, for her very high research standard, which significantly improve the quality of this thesis.

Thank Professor Andrew Horner, one of the most renowned computer music researchers, for his insightful reviews and comments on this thesis, which really boost it to another level.

I would like to express my deepest gratitude to Qing, for her everlasting supportive and positive point of view towards everything I have done. I truly enjoy all those fruitful discussions about music industry, songwriting, hip-hop

and jazz. Her passion about music technologies always makes me believe that what I have been doing is worthwhile.

I am indeed grateful to be living in the Graduate House, where I gradually improve my spoken English, perfect my accompaniment skills, and learn jazz. My life would have been dry and bored without my house friends. Thank my songwriting partner, the real piano accompanist Jonathan de la Cruz. He plays wonderful gigs, sing-along sessions, and always allows me to watch and learn from his sophisticated techniques. Thank Yi Eun, the real musician, for his sincere appreciation of my music. Thank Kelvin, the real singer, for his one of the best vocals in the world. Thank Tianyin, for her nice small gifts, Faichuns, and cookings. Thank JT, for training me to be a better "drinker". There are a lot more people I want to thank: Billy and Aimie, Ben and Haidi, Chae Yin, Maggie, Wenpin, Haoyuan, Dinghua, Tewei, Lili, Shengda, Pingyu, Xiaowan, Dandan, Jay Lu, the music group, the card game group, the fitness group, the dancing group, the list can go on and on... I am just so reluctant to leave and start a new life.

My sincere thanks also go to my labmates in Chow Yei Ching Building 101. Here I met Ho-Cheung, with whom I share a lot of life stories. I met Bony, Dominic, and Sam. They are always willing to help, welcoming to talk, and passionate to share all kinds of ideas. I would also like to thank Xing. He is one of the best colleagues I have in EEE with signal processing and machine learning background. He makes me feel that I am not alone. Finally, I am always grateful to Xiangyu during the teaching assistant time. He is humble, hardworking, strong-minded and he never gives up.

I also thank Yuheng, my best friend in Guangzhou, to always have my back when I am in trouble.

Lastly, I take this opportunity to thank my wife, my parents and my close relatives. They are always there when I am in need and they always understand

me. I am greatly indebted to them, for not being able to be around, and not always doing what they expected. For everything that I have done wrong, I am deeply sorry.

Thank you for all the memories in these years:

If I should live forever

And all my dreams come true

My memories of love will be of you

— John Denver

DISCARD THIS PAGE

Table of Contents

Page

Page

Page

DISCARD THIS PAGE

List of Tables

DISCARD THIS PAGE

List of Figures

Figure Page

Figure Page

Nomenclature

ACE	automatic chord estimation
ACQA	Average Chord Quality Accuracy
ASR	automatic speech recognition
BLSTM	bidirectional long short-term memory
BRNN	bidirectional recurrent neural network
CD	contrastive divergence
CNN	convolutional neural network
CQT	constant-Q transform
CRF	conditional random field
CSR	chord symbol recall
CV	cross validation
DBN	deep belief network
DFT	discrete Fourier transform
DNN	deep neural network
DYBM	dynamic Bayesian network
FCNN	fully-connected neural networks
GMM	Gaussian mixture model
HMM	hidden Markov model
HPCP	harmonic pitch class profile
ICA	independent component analysis
LSTM	long short-term memory
LV	large vocabulary
MIR	music information retrieval
MIREX	music information retrieval exchange
MLN	Markov logic network
MLP	multi-layer perceptron
NNLS	non-negative least square
PCA	principle component analysis
PCD	persistent contrastive divergence
PCP	pitch class profile
RBM	restricted Boltzmann machine
RCO	relative correct overlap
RNN	recurrent neural network
SGD	stochastic gradient descent
SQ	segmentation quality
STFT	short-time Fourier transform
SVM	support vector machine
WCSR	weighted chord symbol recall

Abstract

Being well aware of the chord annotation subjectivity issue, this thesis attests the necessity of large vocabulary with a joint argument of machine musicianship and the Turing test. Built upon this premise, it proposes two deep learning based system frameworks that lead to potential practical solutions to large vocabulary automatic chord estimation.

The first framework separates chord segmentation and classification into two tasks, which is unlike all previous approaches that combine them in one single pass. Several deep learning models are implemented and tested. Under the large vocabulary evaluation, the recurrent neural network model shows great potential in balanced performances across different chords. This framework has shown its advantages over large vocabulary evaluation in the automatic chord estimation task of music information retrieval evaluation exchange 2016.

The second framework incorporates a skewed class distribution sensitive approach. It employs an "even chance" scheme to boost the uncommon chords' exposure when training a recurrent neural network sequence decoder. The main drawback of this approach is the low segmentation quality. Nevertheless, it demonstrates the even chance training scheme to be effective for the large vocabulary automatic chord estimation.

Finally, a preliminary study has been conducted for automatic jazz chord estimation. Upon this study, a chord-scale estimation system is built and some semi-automatic or fully automatic jazz improvisation demos are created.

Chapter 1

Introduction

This chapter introduces automatic chord estimation (ACE) by defining the problem in Section 1.1 and motivating it in Section 1.2. Then the thesis structure will be elaborated in Section 1.5, followed by the thesis contributions and the author's list of publications in Section 1.6.

1.1 Problem Definition

Automatic chord estimation, or ACE, within the context of this thesis, refers to a task that **estimates, recognizes, or transcribes the segmented chord sequence from a piece of audio under equal-temperament tonal music constraint** (see Chapter 2 for more definitions on these musical terms). In this definition, the verbs "estimate", "recognize" or "transcribe" all refer to the same process. This process analyzes the input audio, uncovers the underlying chord progression, and segments the audio in terms of these chords. Besides, ACE also asks for the **classification of chord and non-chord (i.e., silence, natural sound, environmental noise, etc.) materials**. Figure 1.1 illustrates ACE as a music information retrieval (MIR) task.

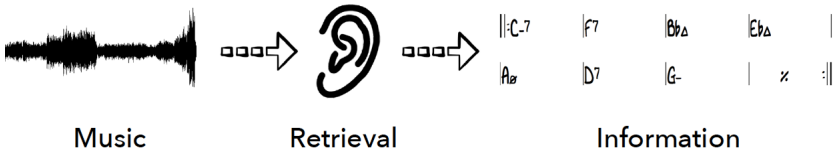

Figure 1.1 ACE as an MIR task. An ACE algorithm (**retrieval**) takes the input audio (**music**), and generates a piece of segmented chord/no-chord sequence (**information**) as output.

1.2 Motivation for ACE

ACE has been one of the most important problems in MIR. The motivation for ACE research is four-fold, explained as follows:

1.2.1 ACE as submodule to other tasks

ACE could be a subproblem of other tasks such as: cover song identification [Bel07, Lee06b, SGH10], which makes use of chord sequences similarity; music structural segmentation [BP05], where structure is cued by repetitions of chord sequences; and genre classification [CYL+08, PSRI09], where the chord sequence itself carries genre information. It could also play a critical role in problems such as: audio key detection [PT12, PM10], where the estimated chord sequence could be used to infer the key or key sequence; and downbeat estimation [PP08, MD10b], where the chord boundaries and downbeats sometimes overlap.

1.2.2 ACE as chord transcription engine

For human beings, recognizing and transcribing chords from audio, especially the ability to distinguish among similar chords (e.g., $C7$, $C9$ and $C7/G$), is a way to demonstrate sophisticated musicianship. People with chord transcription abilities have developed the popular chord-lyrics websites such as

UltimateGuitar [1], E-chords [2] and many others [3] alike, where the chords of millions of songs can be found. To be useful for practical purpose (e.g., song covering, rehearsal, performance, busking), those chords are often captured in great detail, with a very large vocabulary including the suspensions, extensions, inversions and alterations. The annotators try to recover every subtle flavor of the original recordings by means of the handy chord labels. Figure 1.2 showcases some of the human chord annotation examples from the above mentioned websites.

Figure 1.2 Human chord annotation examples

However, musical sophistication cannot be easily replicated. As a result, at some point, the need for chord annotations will overwhelm the human annotation workforce. With the ever increasing music production rate [IFP16], it is foreseeable that the future chord-lyrics services will increasingly rely on

[1] `ultimate-guitar.com`
[2] `e-chords.com`
[3] `polygonguitar.blogspot.hk`; `chords-haven.blogspot.hk`; `azchords.com`

the ACE technologies. Consequently, a chord transcription engine powered by large vocabulary ACE (or LVACE) technology will become necessary.

1.2.3 ACE as part of artificial musicianship

Human musicianship requires as least one of the four kinds of trainings: ear training, sight-reading, composition and improvisation (Figure 1.3) . MIR researches mostly focus on the "ear training" and "sight-reading". The ability to transcribe "chords" is obviously a result of "ear training".

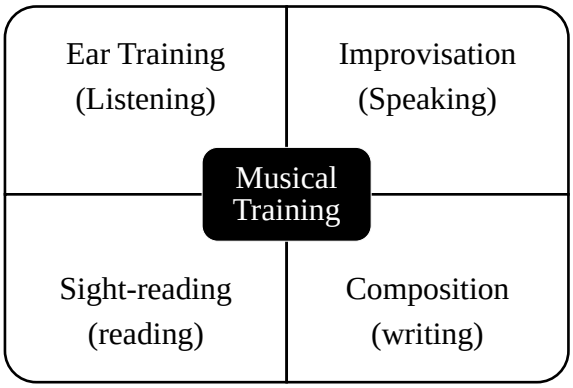

Figure 1.3 Musical training v.s. language training

Ear training, according to the EarMasterPro [4] (as elaborated in Figure 1.4), includes several sub-trainings. The "identification of chords" is one of them. Therefore, in artificial musicianship, ACE plays an important part of the artificial musical "ear".

[4] `http://www.earmaster.com/`

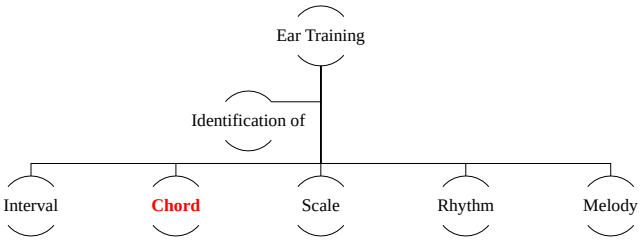

Figure 1.4 Ear training according to the EarMasterPro

1.2.4 ACE inspires scientific researches on human audition and mind

Last but not the least, ACE research, in an algorithmic level, could shed light on the working mechanism of human's perception towards musical harmony. This similar motivation is shared by many other artificial intelligent researches[LB95, HDFN95], where the explorations of algorithmic or machine learning solutions themselves inspire the scientific researches on the visual perception system within the human brain.

1.3 Motivation for LVACE

Now that the motivation for ACE is clear, the motivation for LVACE could be reduced to the necessity of LV. If an ACE system does not support LV (for example, only *major* and *minor* chords are supported), it misses a lot of harmonic details of the original song's arrangement, thus its output will not be suitable for practical usage such as song covering, rehearsal, or busking. Furthermore, the absence of LV will make the system impossible to pass the Turing test since any human chord annotation expert is able to label chords with a large vocabulary. Therefore:

- LV is necessary for a practical chord transcription service.
- LV is necessary for an ACE system to pass the Turing test [Tur50].

1.4 Research Scope

It is necessary to firstly define the scope of this study:

- What types of large vocabularies will be studied?
- What types of music will be studied?

In Chapter 2, we will introduce several vocabularies, including the *Sevenths-Bass* vocabulary, which is used by the current standard ACE evaluation. This is regarded as a large vocabulary, as oppose to the one only containing *major* and *minor* triads. The *SeventhsBass* will be the main focused of this thesis.

Besides, pop and rock music will be the main focus of this thesis, because almost all ACE datasets are built around these two genres. In Chapter 5 we try to extend it to jazz, but nevertheless, most of the discussions and conclusions in this thesis are only verified under pop and rock music.

1.5 Thesis Structure

This thesis focuses on deep learning solutions to LVACE. The main contributions of this thesis can be found in Chapter 3, 4 and 5, where Chapter 3 and 4 are parallel sections presenting two different LVACE solutions, and Chapter 5 takes one of them and build algorithmic music systems for jazz. Concretely:

Chapter 1 — is the introduction to the whole thesis. It defines the ACE problem, motivates the LVACE from different perspectives, and sketches the thesis outline and contributions.

Chapter 2 — is the literature review. The chapter firstly provides the necessary musical fundamentals for a good understanding of ACE. Then it looks back into ACE's history and tries to come up with a storyline of different ACE modules and submodules. Finally it examines various preferences on ACE evaluations, especially the issues and arguments on LVACE evaluation and human annotation subjectivity.

Chapter 3 — describes an LVACE approach that makes hybrid use of a classical segmentation process and a deep learning based chord labeling process. The chapter explores variations and limitations of this system framework, and gives much analysis of the system behaviors under different settings.

Chapter 4 — describes an LVACE approach where the sequence segmentation and classification are all done by a single recurrent neural network. To accommodate to large vocabulary, a skewed classes oriented training scheme is used. This training scheme is shown to be very effective in improving the system's vocabulary versatility.

Chapter 5 — extends the application of LVACE to jazz music. The chapter firstly introduces the musical fundamentals of jazz. Then it extends the LVACE approaches to jazz, and brings in an extra "scale" estimation process to enable the "chord-scale" estimation. Combining with two jazz improvisation

interfaces, it shows how novice users can make good improvisations out of jazz backings.

Chapter 6 — concludes the thesis with suggestions to possible future directions of ACE and LVACE. At the end of this chapter we speculate on the feasibility of an LVACE system for all chords in all kinds of music.

1.6 Contributions and Publications by the Author

The major contributions of this thesis are mainly on the deep learning methods and skewed class distribution oriented techniques for LVACE. Particularly, this thesis proposes two LVACE approaches:

1. A hybrid classical segmentation and deep neural nets chord labeling approach;

2. A recurrent neural network sequence decoding with an even chance training scheme approach.

There are also minor contributions in the algorithmic applications of jazz. Particularly, this thesis proposes:

1. A jazz chord-scale estimation system that augments the LVACE framework with a local scale tracking algorithm.

2. A fully automatic jazz improvisation process that combines the chord-scale estimation system and a note generation process.

3. Two semi-automatic jazz improvisation platforms that enable users to create melodies based on a chord-scale sequence.

In correspond to the thesis contributions, the author has published the following articles:

- Deng, J., Kwok, Y. K., Large Vocabulary Automatic Chord Estimation with an Even Chance Training Scheme, In Proceedings of the 18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017

- Deng, J., Kwok, Y. K., A Hybrid Gaussian-HMM-Deep-Learning Approach For Automatic Chord Estimation with Very Large Vocabulary, In Proceedings of the 17th International Society for Music Information Retrieval Conference, New York City, USA, 2016

- Deng, J., Kwok, Y. K., A Chord-scale Approach to Automatic Jazz Improvisation, In Late-breaking/demo Proceedings of the 17th International Society for Music Information Retrieval Conference, New York City, USA, 2016

- Deng, J., Kwok, Y. K., Automatic Chord Estimation on SeventhsBass Chord Vocabulary Using Deep Neural Network, In Proceedings of the 41st International Conference on Acoustics, Speech, and Signal Processing, Shanghai, China, 2016

- Deng, J., Lau, F. C. M., and Kwok, Y. K., ArmKeyBoard: A Mobile Keyboard Instrument Based on Chord-Scale System and Tonal Hierarchy. In Proceedings of the 40th International Computer Music Conference, Athens, Greece, 2014.

- Deng, J., Lau, F. C. M., Ng, H. C., Kwok, Y. K., Chen, H. K., and Liu, Y. H., WIJAM: A Mobile Collaborative Improvisation Platform under

Master-Players Paradigm. In Proceedings of the 14th International Conference on New Interfaces for Musical Expression, London, UK., 2014

There are currently two articles under review:

- Deng, J., Kwok, Y. K., Large Vocabulary Automatic Chord Estimation Using Deep Neural Nets: Design Framework, System Variations and Limitations, submitted to EURASIP Journal on Audio, Speech, and Music Processing, 2017 (under review)

- Deng, J., Kwok, Y. K., Large Vocabulary Automatic Chord Estimation Using Bidirectional Long Short-Term Memory Recurrent Neural Network with Even Chance Training, submitted to Journal of New Music Research, 2017 (minor revision, can be accepted after one more round)

Chapter 2

Background and Related Work

This chapter introduces the background and related works of the ACE research. Section 2.1 familiarizes the readers with the necessary musical fundamentals for understanding ACE. Section 2.2 examines an exhaustive list of ACE literatures that reveal the evolution and variation of ACE approaches in various aspects. Afterwards, Section 2.3 will give an overview of the ACE evaluation methods. The chapter will be summarized and concluded in Section 2.4 with the current research gaps leading to the main content of the thesis.

2.1 Musical Fundamentals

This section introduces a necessary amount of musical concepts. On one hand it serves as the underlying basis of an ACE system, and on the other hand it facilitates the readers' understanding of the theoretical underpinnings.

2.1.1 Basic Concepts

Music can be understood as an organized sound sequence composed of pitches, noise and silence. The art of music lies very much in the design and arrangement of pitches. Therefore it is of principle need to understand the

concept of pitch, upon which we can further understand other related concepts that gradually lead to the concept of chord.

Pitch — *Pitch*, by definition [Ran99], is:

> The perceived quality of a sound that is chiefly a function of its fundamental frequency - the number of oscillations per second (called *Hertz, abbr. Hz) of the sounding object or of the particles of air excited by it.

According to this definition, pitch can be understood as a subjective measure of a sound's *fundamental frequency*. Although subjective, people usually use pitch to actually refer to the fundamental frequency itself. In this thesis, unless otherwise clarified, the terms *pitch* and *fundamental frequency* are used interchangeably. Sometimes pitch can also be substituted with *tone* [Ran99], which means: "a sound of definite pitch; a pitch."

A pitch may not only contain its fundamental frequency, but also a *harmonic series*, or *harmonics*, which is [Ran99]:

> In acoustics, a series of frequencies, all of which are integral multiples of a single frequency termed the fundamental.

For example, when a guitar string is pluck, it generates a pitch with harmonics, and this is mainly because of the physical standing wave [HE09] phenomenon.

Note — Pitch and note are closely related. *Note*, by definition [Ran99], is:

> A symbol used in musical notation to represent the duration of a sound and, when placed upon a staff, to indicate its pitch; more generally (especially in British usage), the pitch itself.

It is easily understood that *note* is a symbol of pitch and its duration, but as a generally acceptable usage, it can also be referred to the pitch itself. With the concept of note, it is not difficult to understand the concept of interval.

Interval — *Interval*, by definition [Ran99], is:

> The relationship between two pitches. For purpose of Western tonal music, intervals are named according to ... the number of semitones (the smallest interval in the Western system) between the two pitches.

The interval between two pitches is usually measured as the their ratio (i.e., the ratio of their fundamental frequencies).

Equal Temperament — For any tonal music system, there must be a kind of *temperament* that defines the relationship among different pitches that an instrument produces. Well known temperaments include Pythagorean, just intonation, mean-tone, well temperament and equal temperament. Interested readers can refer to [Bar04] for more information on this topic. Within all these temperaments, an *octave* is defined as an interval where the ratio of two pitches is 2:1.

The most dominant temperament in modern music styles, including the modern pop, rock, jazz and many other styles, is the *equal temperament*, particularly the *twelve-tone equal temperament*. In this temperament, an octave contains 12 equally-sized semitones, each has an interval of $\sqrt[12]{2}$.

Within a 12-tone-equal-tempered octave, if we name the first pitch as C, then the pitch sequence can be normally named as: C, $C\#/Db$, D, $D\#/Eb$, E, F, $F\#/Gb$, G, $G\#/Ab$, A, $A\#/Bb$, B where "/" means "or", "#" (spelling

"sharp") means with one semitone higher, and "b" (spelling "flat") means with one semitone lower. The next pitch of this sequence will be an octave above the original C. To differentiate octaves, in scientific pitch notation [fS75] an Arabic number is usually appended to the pitch names to indicate the octave heights, such as $C4$, $A3$, $C5$.

Pitch Class and Octave Equivalence — "A pitch without reference to the octave or register in which it occurs" is a *pitch class* [Ran99]. Humans are reported to have the ability to perceive pitch class. This perceptual phenomenon is called *octave equivalence* [Ran99, Bor42]:

> The feature of musical perception according to which all pitches separated by one or more perfect octaves are regarded as belonging to the same class or as being in some sense equivalent.

Tuning — Although an equal temperament can be set with any pitch sequence that fulfills the requirement, there is technically no constraint on pitch frequencies. For example, the note $B5$ can be of $1000\,Hz$, or $2000\,Hz$, or just any number like $23.75892\,Hz$, as long as the sequence is equal-tempered. Without a proper frequency constraint, any two equal temperament instruments have almost no chance to be in tuned. In order to get the instruments "in tuned", there shall be some kind of frequency constraint, or "tuning". *Tuning*, by musical definition [Ran99], is:

> The act of adjusting the fundamental sounding frequency or frequencies of an instrument, usually in order to bring it or them into agreement with some predetermined pitch.

A standard tuning normally adopted in tonal music is $A4 = 440\,Hz$, where $A4$ stands for the A note above $C4$ (or the middle C), which could be illustrated as the fourth C key on a standard 88-key piano keyboard.

Intervals within an Octave — All the music objects considered in this thesis are assumed to be in 12-tone equal temperament, and very close to the standard tuning. Therefore it is very useful to study a few common intervals within the 12-tone temperament system.

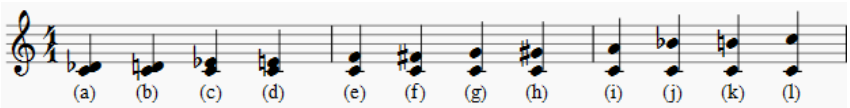

Figure 2.1 Music interval examples

Figure 2.1 shows a list of intervals and their short notations. All intervals are built upon $C4$, denoted by the note on the extra line below the staff. These intervals are:

(a) minor second - m2 (1)

(b) major second - M2 (2)

(c) minor third - m3 (3)

(d) major third - M3 (4)

(e) perfect fourth - P4 (5)

(f) augment fourth - A4 / diminished fifth - D5 / tritone - TT (6)

(g) perfect fifth - P5 (7)

(h) minor sixth - m6 (8)

(i) major sixth - M6 (9)

(j) minor seventh - m7 (10)

(k) major seventh - M7 (11)

(l) perfect octave - P8 (12)

The number of semitones is indicated in the bracket next to the interval name. The interval that contains 0 semitone is called *unison* or *perfect unison* (P1), meaning that the two pitches are equal in fundamental frequencies.

2.1.2 Chords and Chord Progressions

This subsection elaborates the concept of chord and chord progression based on the previously introduced concepts.

Chord — *Chord* is one of the most important concepts in this thesis, which studies automatic chord estimation technologies. *Chord*, by definition [Ran99], is:

> Three or more pitches sounded simultaneously or functioning as if sounded simultaneously; two such pitches are normally referred to as an interval.

All chords considered in this thesis are under equal temperament. In this sense, a chord can be decomposed into several stacked intervals. As the above music dictionary entry [Ran99] further elaborates:

> In the analysis of tonal music, all chords may be regarded as consisting of or deriving from two or more thirds (whether major or minor) arranged one above another (e.g. G-B-D-F).

A chord with three pitches is called a *triad*, with four pitches a *tetrad*. A triad is composed of two intervals, while a tetrad is composed of three. Assuming octave equivalence, Table 2.1 presents five frequently used triads and tetrads with examples shown in terms of pitch classes (chord tones). It

Triad/Tetrad	Chord	Short symbol	Stacked intervals	Example in chord tones
Triad	major	maj	M3 + m3	Cmaj: C-E-G
Triad	minor	min	m3 + M3	Dmin: D-F-A
Tetrad	seventh	7	M3 + m3 + m3	E7: E-G#-B-D
Tetrad	major seventh	maj7	M3 + m3 + M3	Fmaj7: F-A-C-E
Tetrad	minor seventh	min7	m3 + M3 + m3	Amin7: A-C-E-G

Table 2.1 Chord examples

should be noted that every chord has a *root* [Ran99]

> (A root is,) in tonal harmony, the fundamental or generating pitch of a triad or chord. If the pitches of a chord are arranged as a series of superimposed thirds, the lowest pitch is the root.

For example, a $Fmaj7$ chord has a root F, while $maj7$ is the chord's *quality*, which is independent of the root. Root should be distinguished from *bass*, which is the "lowest pitch of any single chord" [Ran99]. The *root* concept has a symbolic meaning, while the *bass* concept has a perceptual meaning. Two other important concepts can help clarify their relationship [Ran99]:

- A chord is in *root position*, if its *bass* is the same as its *root*;
- Otherwise, a chord is an *inversion*

For example, Figure 2.2 shows three possible positions of a $Cmaj$ chord. The first one satisfies the requirement of a *root position*, but the other two do not. The second one starts the chord with E rather than C. This is called the *first*

inversion, since its bass is the first pitch class next to the root in the original chord tone sequence. The third one starts with G, and this is called the *second inversion*. As for the major chord, a shorthanded notation omits the "*maj*"

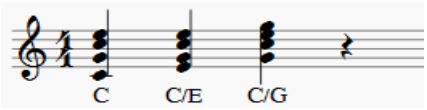

Figure 2.2 C, C/E and C/G

and only keeps the root symbol. An inversion is labeled as the original chord symbol concatenated by its bass pitch class, bridged with a slash symbol "/". Thus these three instances in Figure 2.2 are labeled as C, C/E and C/G respectively. Since inversions have their own labels, we sometimes consider them as standalone chords.

Chord Progression — A *chord progression* or *harmonic progression* is [Sch89]:

> a series of musical chords, or chord changes that "aims for a definite goal" of establishing (or contradicting) a tonality founded on a key, root or tonic chord.

The *tonality* here means [Ran99]:

> In Western music, the organized relationships of tones with reference to a definite center, the tonic, and generally to a community of pitch classes, called a scale, of which the tonic is the principal tone; sometimes also synonymous with *key*.

Key and Scale — Particularly, a *key* is [Ran99]:

> In tonal music, the pitch relationships that establish a single pitch class as a tonal center or tonic (or key note), with respect to which the remaining pitches have subordinate functions.

There are *major key* and *minor key*, characterized by their own *major scale* and *minor scale*. Defining a *whole step* (W) as an interval containing two consecutive semitones, and a *half step* (H) as containing one semitone, the two scales can be expressed as two ordered lists:

$$\begin{aligned} \textit{major scale} &= (W, W, H, W, W, W, H), \\ \textit{minor scale} &= (W, H, W, W, H, W, W), \end{aligned} \tag{2.1}$$

The keys and scales can be named after the starting pitch class, such as "C major key" and "D minor key"; and the scales "C major scale" and "D minor scale". These seven-pitch scales, containing two half steps and five whole steps, are called *diatonic scales*, regardless of the starting pitch class. Chord progressions that built upon these scales are decisive clues to the underlying keys. A *diatonic chord progression* is a sequence of seven chords built on a diatonic scale. For example, in C major key, such progression can be:

$$Cmaj - Dmin - Emin - Fmaj - Gmaj - Amin - Bdim, \tag{2.2}$$

Short Summary — For now, we only need to know that a *chord progression* is a "progression" of chords, or sequence of chords that describes or decides the harmonic structure of a piece of music. Figure 2.3 illustrates a few sample chord progressions. Note that a minor chord can be shorthand notated as "m" (not to be confused with the minor third interval notation). Although chords and chord progressions are essential in all kinds of tonal music, the notations

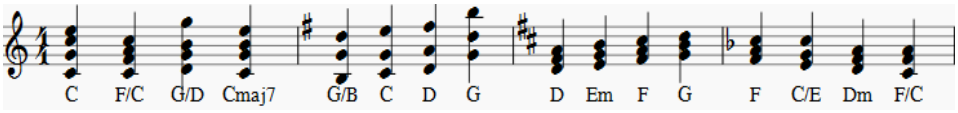

Figure 2.3 Example chord progressions in key of C, G, D and F

of them are mainly prevalent in pop, rock, jazz and many other music styles that belong to a more popular culture.

2.1.3 Sheet Music, Lead Sheets and Chord-lyrics

Sheet music is a form of music notation. It records all note level and some expression level elements within a piece of music with symbols. It is often written or printed on a staff with musical notes and expression marks. It is particularly suitable for solo instrumental pieces or orchestration pieces. When it is used to notate a song, the full musical arrangement will usually be reduced to a simple piano accompaniment in order to fit into a piece of sheet music.

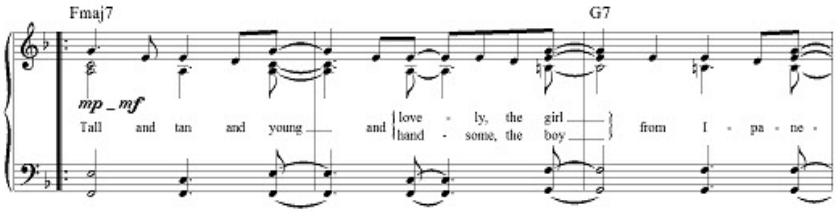

Figure 2.4 Sheet music - The Girl From Ipanema

Jazz music is usually notated on a *lead sheet*. It is a type of sheet music with chords, lyrics, and melody. The accompaniment to the melody is ad-lib rendered from the chord symbols by the musician.

Other styles of songs can also be notated on a lead sheet. But since most people are already so familiar with the melodies before they get access to the sheets, the melody part of the lead sheet can sometime be omitted and

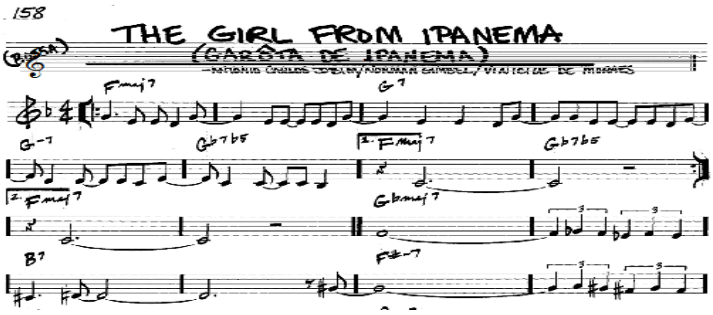

Figure 2.5 Lead sheet - The Girl From Ipanema

become just chords and lyrics. In this thesis such kind of notations are called *chord-lyrics*.

```
FM7
Tall and tan and young and lovely

     G7
The girl from Ipanema goes walking,

    Gm7                    F#7                    FM7
And when she passes, each one she passes goes, "Ahhh."
```

Figure 2.6 Chord-lyrics - The Girl From Ipanema

Figure 2.4 to 2.6 shows three excerpts of sheet music [1], lead sheet [2] and chord-lyrics [3] of the same song.

[1] www.onlinesheetmusic.com
[2] www.sheetmusicdirect.com
[3] www.traditionalmusic.co.uk

2.2 Review of ACE Design

ACE is an algorithmic process that automates the chord transcription of a piece of tonal music. This process includes the identification of chord boundaries and the recognition of chord labels. This section reviews the ACE literatures. First of all, the general system framework is introduced in Section 2.2.1. Then various feature extraction and feature learning methods are reviewed and compared in Section 2.2.2 and 2.2.3. This is followed by a reexamination of different pattern recognition techniques in Section 2.2.4 and 2.2.5. At last, Section 2.2.6 raises the issue of chord vocabularies, which will focus on several arguments for the necessity of incorporating a large vocabulary and chord inversions into an ACE system.

2.2.1 General System Framework

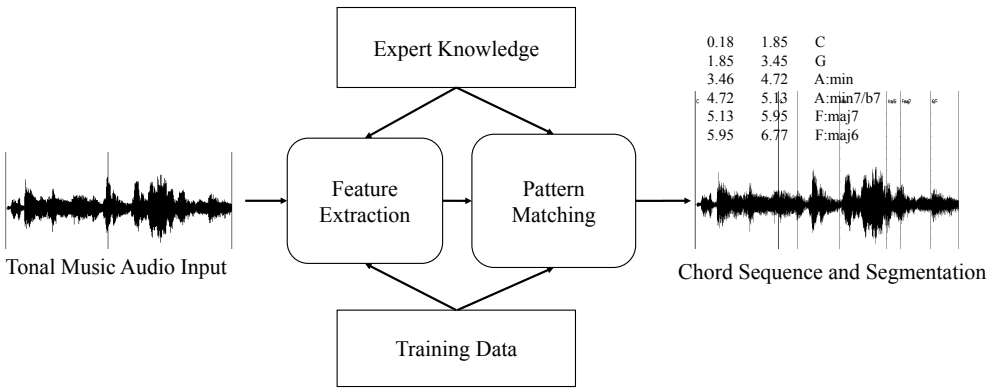

Figure 2.7 General system framework of ACE

The general work-flow of an ACE system is depicted in Figure 2.7. Similar to other pattern recognition systems [DHS12], such as the *automatic speech*

recognition (ASR) [HAHFBR01] system, it contains two big functional modules - feature extraction and pattern matching, both can be trained using ground truth data or powered by expert domain knowledge. Different from the working domains of other applications, ACE takes a piece of (chordal) tonal music audio as input, and exports a time-segmented chord sequence. The "tonal music" nature of the input and output imposes unique constraints to the two functional modules. As a result, their implementations will have their own characteristics.

In order to better understand ACE, we distinguish between the basic problem settings of ASR and ACE, both being roughly recognized as audio-text transcription problems. Table 2.2 presents two fundamental differences between ASR and ACE. Note how the "segmentation" requirement of ACE fun-

	ASR	ACE
audio input	speech	tonal music
text output	word sequence	chord sequence with segmentation

Table 2.2 I/O of ASR and ACE

damentally separates the two otherwise similar problems. Meanwhile, the prior knowledge that the chords are often segmented periodically/rhythmically allows ACE a more flexible system design methodology.

2.2.2 Feature Extraction

Feature extraction is the first step of an ACE system. In this section, the fundamental ACE feature extraction paradigm will be firstly introduced, and then its different variants will be reviewed in terms of different practical concerns.

Pitch Class Profile, Chroma and Chromagram — The first known ACE research is published in 1999 by Fujishima [Fuj99]. The ACE algorithm firstly "transforms a fragment of the input sound stream to a DFT spectrum", where DFT stands for *discrete-Fourier-transform*. The spectrum is then summarized into a *pitch class profile* (PCP), which is "a twelve dimension vector which represents the intensities of the twelve semitone pitch classes". PCP is calculated as follows:

$$PCP(p) = \sum_{s.t.\ M(l)=p} ||X(l)||^2, \tag{2.3}$$

where p is a pitch class, $X(l)$ is the DFT spectrum, and $M(l)$ is a table that maps the linear N-point DFT frequencies to the non-linear pitch class frequencies:

$$M(l) = round(12 \times log_2(f_s \cdot \frac{l}{N})/f_{ref}) \, mod \, 12, \tag{2.4}$$

where $l = 1, 2, ..., N/2-1$, f_s is the sampling frequency, and f_{ref} is the reference frequency that corresponds to $p = 0$. Repeat this process for every consecutive "fragment" of the input, it actually becomes a "*short-time-Fourier-transform* (STFT) - PCPs" feature extraction process, which outputs a sequence of PCPs.

As a matter of fact, Fujishima's PCP is formally discussed much earlier in 1964 by Shepard [She64], who proposes the concept of *chroma*. A chroma "transforms frequency into octave equivalence classes", as pointed out by Wakefield [Wak99], who in the same paper invents the term *chromagram* to "extend the concept of chroma to include the dimension of time". As a result, one can relate PCP directly to chroma, and a sequence of PCPs to chromagram.

Except for a few works that apply wavelet [SJ01] transform, or autocorrelation [BMS+00, ZR07] for pitch tracking, the "STFT-chromagram" is the

dominant feature extraction paradigm that almost all subsequent ACE approaches resemble in various degrees.

Musical Concerns — Each DFT spectrum is linear spaced in frequency. But as recapped in Section 2.1, musical pitches are arranged in terms of the ratios of their fundamental frequencies. Therefore equal-tempered pitches are arrange linearly only within a log scale. This is illustrated in Figure 2.8. Therefore,

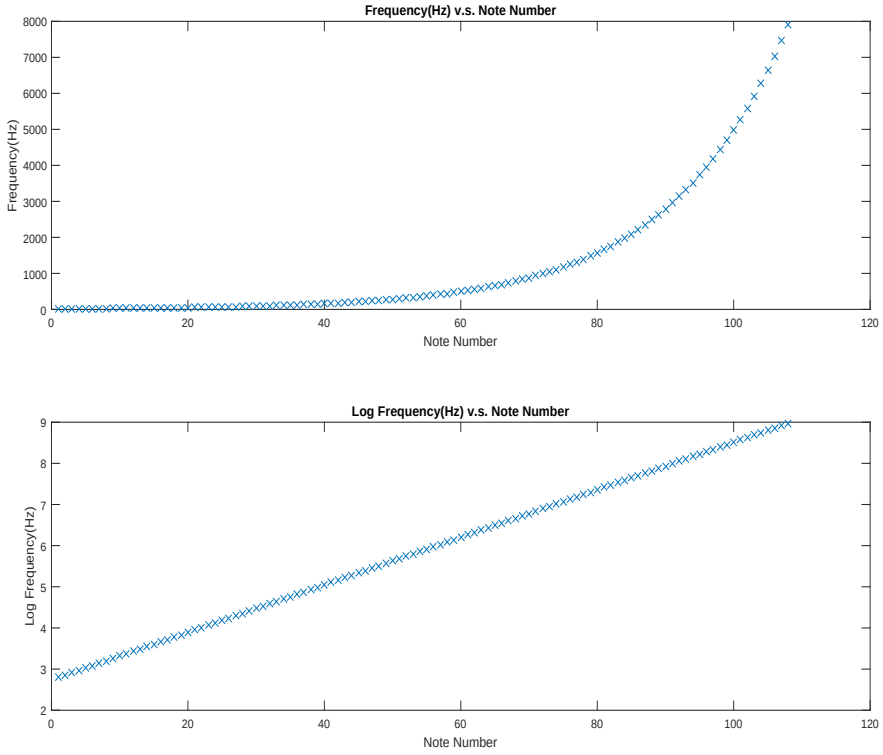

Figure 2.8 Relationship between equal temperament note and frequency (note number from C0 to B8, totally 108 notes)

STFT leads to over-sampling of high frequency pitches, and under-sampling of low frequency pitches. This is because DFT transforms a time-domain signal

with a fixed-size window over all frequencies [OWN83], overlooking that the low frequency signals occupy longer time period to establish cycles.

A "musical friendly" variant of DFT is proposed in 1991 by Brown [Bro91], who uses variable length windows to capture a constant "Q" cycles in different frequencies:

$$X(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} W(k,n)x(n) \exp\{-j2\pi Qn/N(k)\}, \tag{2.5}$$

where Q is a constant, $X(k)$ is the k^{th} constant-Q spectrum bin, $N(k)$ is the k^{th} window size, $W(k,\cdot)$ is the k^{th} window and $x(n)$ is the n^{th} input sample. This process is called *constant-Q transformation* (CQT).

The first ACE system to use this technique is published by Nawab et al. [NAW01], who use the constant-Q spectra as a pool for peak picking fundamental frequencies of different chords. Later Bello and Pickens [BP05], Harte and Sandler [HS05] start to apply this in a more straightforward context, where they compute a chroma (Harte and Sandler also name it *harmonic pitch class profile* (HPCP)) directly from the constant-Q spectrum as:

$$C(b) = \sum_{m=0}^{M} |X(b+mB)|,\ b \in [1,B], \tag{2.6}$$

where $C(\cdot)$ is the chroma, $X(\cdot)$ is the constant-Q spectrum and B is the total number of chroma bins. Depending on the value of B, the chroma may have different pitch resolutions. A common value of B is 36 [BP05, HS05, Oud10, RUS+09, WSDR09, HB12, Cho14], corresponding to 3 bins per semitone, but other values such as 48, 96 or even not multiple of 12 are also possible.

To solve a similar problem that CQT does, Mauch instead proposes a "log-frequency spectrum" approach. He starts by noticing that [Mau10]:

> The main problem in mapping the spectrum to a log-frequency representation is that in the low frequency range several log-frequency bins may fall between two DFT bins, while in high frequency regions the reverse is true.

Thus his algorithm upsamples the DFT representation first, and then downsamples it again to a log-frequency scale. In order to perform the upsampling and downsampling, the algorithm uses two cosine interpolation kernels. The first kernel is:

$$h(f, f_i) = \begin{cases} \frac{1}{2} cos(\frac{2\pi(f-f_i)}{f_s/N_F}) + \frac{1}{2}, & \text{if } |f_i - f| < f_s/N_F \\ 0 & \text{otherwise,} \end{cases} \tag{2.7}$$

where N_F is the DFT frame length and f_i is the i^{th} digital frequency of the original spectrum. f is a sequence of digital frequencies that is spaced 40 times more intensive than f_i, or it has 1/40 of the original DFT's frequency resolution $\delta f (\delta f = f_s/N_F)$. The upsampled spectrum M_f is calculated by:

$$M_f = \sum_{i=0}^{N_F} h(f, f_i) X_i, \tag{2.8}$$

where X_i is the i^{th} DFT bin. The second kernel is:

$$h_l(f, f_k) = \begin{cases} \frac{1}{2} cos(\frac{2\pi(f-f_k)}{\delta f(f)}) + \frac{1}{2}, & \text{if } |f_k - f| < \delta f(f) \\ 0 & \text{otherwise,} \end{cases} \tag{2.9}$$

where f_k is the k^{th} downsampled digital frequency, and $\delta f(f)$ is a constant-Q function of f. Concretely, it is:

$$\delta f(f) = f/Q, \tag{2.10}$$

where in a 36 bins per octave case, as Mauch indicates, $Q = 36/ln2 \approx 51.94$. $\delta f(f)$ makes sure that the upsampled spectrum will be mapped to a log-frequency spectrum as the kernel applies to it:

$$Y_k = \sum_f h_l(f, f_k) M_f, \tag{2.11}$$

where Y_k is the final log-frequency spectrum. This approach, albeit not as theoretically sound as the constant-Q transform, works very well in practice. Combining the two kernels into a single operation, it becomes a linear-log transformation matrix as illustrated in Figure 2.9. Note that the matrix is very sparse.

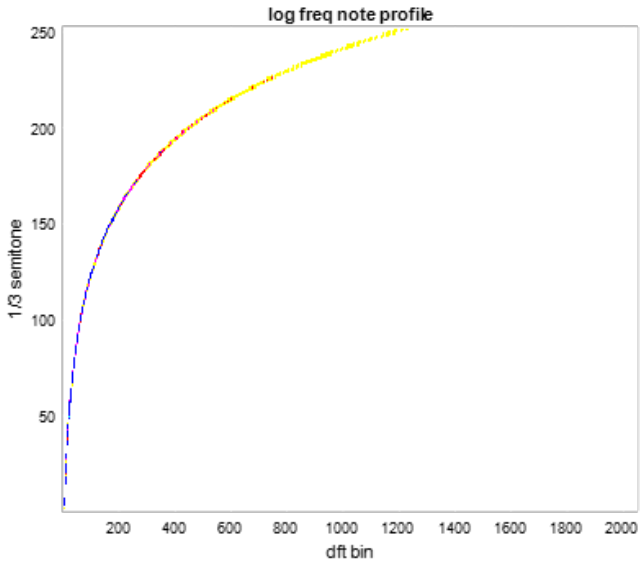

Figure 2.9 Linear-log transformation matrix

To model the "intervals" instead of "pitch classes", and to further reduce the feature dimensions, Harte et al. propose *tonal centroid* [HSG06], which is inspired by Chew's "Spiral Array" [Che00]. It is a point in 6-dimension space, so that when a 12-dimension pitch class is mapped onto it, "close harmonic

relations such as fifths and thirds appear as small Euclidean distances". Technically sound though, only a few authors explore this feature in their ACE systems [LS08, HCB12].

Noise Removal — Another concern is that the noise within an audio will get into the chroma. It can be reduced along the time dimension, such as by smoothing with a mean filter [HS05, HB12] or a median filter [KO09, MD08], which applies to the features across a number frames. Perhaps a more intuitive way to reduce noise from a time perspective is by "beat-averaging"[BP05, MD10a], since the onsets and offsets of chords often coincide with beats [GM99]. Moreover, structural information can also be leveraged [MND09, CB11, Cho14] to reduce noise within the audio features.

Noise can also be alleviated along the frequency dimension in several ways. Catteau et al. [CML07] subtracts from the original log-frequency spectrum a "background spectrum", which is computed by convolving the log-frequency spectrum with an one-octave wide Hamming window. Alternatively, Varewyck et al. [VPM08] computes the background spectrum by "filtering the amplitude spectrum with a median filter". Noticing that the noise often comes from percussive instruments, Reed et al. [RUS+09] attempts to remove noise by doing harmonic-percussive separation [OMLR+08]. This method is also adopted in later works such as [NMSRDB12] and [UUN+10].

Mauch [Mau10] puts forward a "standardization" process that reshapes the spectra along the frequency dimension. In this process, every log-frequency spectrum is first subtracted from its "running mean", similar to Catteau's background spectrum removal; then it is divided by the "running standard", similar to Klapuri's spectral whitening technique [Kla06].

Harmonics Removal — Harmonics could be considered another type of "noise" in ACE. This is because the human auditory system identifies a chord by first identifying the fundamental frequencies of all notes being played. Removing harmonics from the signal is to some extent equivalent to recovering the true note activations, or extracting the fundamental frequencies.

To remove harmonics, Lee proposes the *enhanced pitch class profile* (EPCP) [Lee06a]. In this approach, the DFT spectrum is transformed to a *harmonic product spectrum* (HPS) through the following process:

$$HPS(k) = \prod_{m=0}^{M} |X(2^m k)|, \tag{2.12}$$

where $X(\cdot)$ is the DFT spectrum. In practice this process can reduce a certain amount of harmonics in the original spectrum, resulting in a new spectrum that is more focused on the fundamental frequencies of the activated notes. The EPCP is then computed using equation 2.6.

Ryynanen and Klapuri invent a similar approach [RK08], where "the salience, or strength, of each F0 candidate is calculated as a weighted sum of the amplitudes of its harmonic partials in a spectrally whitened signal frame" (F0 means the fundamental frequency).

A more computationally intensive yet theoretically sound approach is the "approximate note transcription" approach, which is a non-negative-least-square (NNLS) fitting process proposed by Mauch [MD10a]. The process tries to fit the best linear combination of a set of harmonic series profiles to the log-frequency spectrum. Concretely, the log-spectrum Y can be expressed as:

$$Y \approx Ex, \tag{2.13}$$

where E is a dictionary of harmonic series profiles, and x is the note activation pattern to be optimized. Each entry of E is a sequence of geometrically declining overtone amplitudes [Góm06a]:

$$a_k = s^{k-1},\ s \in (0, 1), \tag{2.14}$$

where k indicates the k^{th} upper partial, and a larger s means a slower decline. Normally s is between $[0.6, 0.9]$. In order to find out an x that minimizes the difference between Y and Ex, an NNLS method [LH95] is used.

Tuning Compensation — Despite standardized as $A4 = 440\,Hz$, the actual tuning of musical pieces vary from time to time. On one hand, it is difficult to tune a pitch at a very high resolution, on the other hand, some pieces are actually deviating from the standard tuning deliberately. For example, as pointed out by Harte [Har10], "many of the Beatles songs deviate from this tuning (the standard tuning)".

Human auditory system is not sensitive to the slight detuning. But for a computer system, the detuning needs to be compensated for better performance. Sheh and Ellis [SE03] are among the first to recognize this and use a PCP of 24-bin instead of 12-bin to "give some flexibility in accounting for slight variations in tuning".

Later, Harte [HS05] introduces a tuning compensation algorithm. This algorithm first quadratic interpolates on each 36-bin constant-Q spectrum, peak-picks them across every semitone, and then adjusts the chromagram bins according to the "center tuning value", which is determined as the maximum value of the peaks' histogram. This algorithm is adopted in subsequent works such as [BP05] and [HSG06].

Instead of estimating detuning as a "histogram of peaks", there is anther solution, first proposed by Dressler and Streich [DS07], that interprets detuning as an angle within $[-\pi, \pi)$, where π corresponds to half-semitone. Particularly, Mauch [Mau10] implements a version of this solution within his ACE system, where the amount of detuning is estimated as:

$$\delta = \frac{wrap(-\varphi - 2\pi/3)}{2\pi}, \tag{2.15}$$

where $wrap$ is a function wrapping its input to $[-\pi, \pi)$, and φ is the phase angle at $2\pi/3$ of the DFT of the time averaged log-frequency spectrogram. The tuning frequency is then computed as:

$$\tau = 440 \cdot 2^{\delta/12}, \tag{2.16}$$

and the detuning is thus compensated by interpolating the original N-bin spectrogram Y_i at Y_{i+p}, where:

$$p = (\log(\tau/440)/\log(2)) \times N \tag{2.17}$$

This angle based tuning compensation method is adopted in many ACE works such as [PP07], [PP08], [MD08], [RUS+09] and [NS09].

Bass Information — Human chord annotators sometimes recognize the chord bass before they figure out the chord itself. Therefore it is justifiable for an ACE system to treat high range and low range pitches differently. This leads to special consideration for bass information, which is first discussed by Yoshioka et al. [YKK+04]. Later, Sumi et al. [SIY+08] explore this again by adding a bass probability cue to Yoshioka's model.

Ryynanen and Klapuri [RK08], augment the chroma vector to be 24-bin, which contains a 12-bin low-frequency chroma and a 12-bin high-frequency

chroma, where the bass information is gathered using an explicit bass line transcription process. Mauch and Dixon [MD10a, MD10b] regard a 24-dimension chromagram as the vertical concatenation of a bass chromagram and a treble chromagram, where each of them are computed by weighting with different profiles.

Further Readings — There are a few review articles on ACE feature extractions. Cho et al. [CWB10] explore some common variations in ACE systems, including different pre- and post-filtering techniques mentioned previously. Jiang et al. [JGKM11] analyze the effect of different chroma feature types on two popular pattern matching techniques. McVicar et al. [MSRNDB14] give a detailed technical review of most ACE feature extraction techniques, including various noise reduction and harmonic removal methods.

2.2.3 Feature Learning

All the above discussed techniques are regarded as "feature engineering", in which features are extracted via handcrafted transformations. However, with machine learning, these features can actually be learned from the data [Ben09]. This is called "feature learning" and it has been widely applied in numerous pattern recognition fields such as computer vision [HS06], automatic speech recognition and natural language processing [DY14].

Feature learning targets the intermediate representations instead of the output labels. Thus it can be achieved by unsupervised learning algorithms such as *principle component analysis* (PCA) [Jol02], *independent component analysis* (ICA) [HKO04] and *k-means clustering* [M$^+$67]. It can also be done with deep neural nets such as *restricted Boltzmann machine* (RBM) [HOT06],

sparse autoencoder [Ng11] and *deep autoencoder* (DAE) [Ben09]. In principle, these algorithms or models try to learn a set/dictionary of sparse or overcomplete basis/atoms, so as to maximize the likelihood that their linear combinations can approximate the raw data.

Supervised learning models, such as *fully-connected neural network* (FCNN), *deep belief network* (DBN) [HOT06], *convolutional neural network* (CNN) [LB95], and *recurrent neural network* (RNN) [Elm90], are also capable of learning features. Since these are mostly discriminative models, they rely on the labeled data to learn (except for the DBN, which takes both labeled and unlabeled data). Hence the features they learn should be good for classification purpose, but may not be always sensible for generative purpose.

Here we introduce two important deep neural nets (DNN) that are recently used in ACE for feature learning purpose, and at the same time we elaborate on some frequently mentioned machine learning and neural network terms.

Fully-connected Neural Network — FCNN is the basic model that other neural nets extend from. Note that FCNN is different from *multilayer perceptrons* (MLP) because the classical "perceptron" model [Ros58] is different from the neuron model (e.g., logistic, tanh, relu, etc.) within a modern neural network [RMG$^+$88]. In this thesis, as in many other literatures, FCNN is used to stand for a fully-connected feedforward neural network.

Figure 2.10 shows an FCNN with three hidden layers. Each layer has a set of *artificial neurons*. Each neuron carries out the following procedures:

1. takes the sum of its inputs (including a bias input) weighted by the incoming connections;

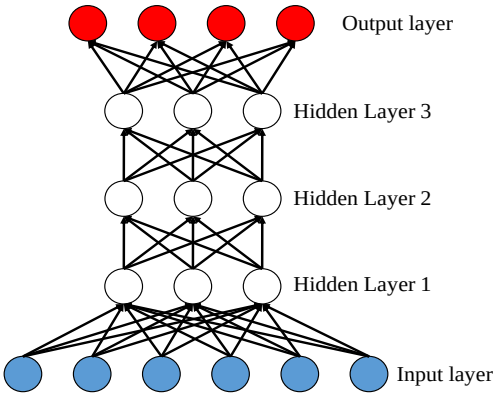

Figure 2.10 fully-connected neural network

2. applies a non-linear function on the weighted sum;

3. generates an *activation* which equals to the output of the above step.

Alternatively, a neuron can be formalized as:

$$y = \sigma(Wx + b), \tag{2.18}$$

where y is the activation, x is the input, W is the weight vector and b is the bias input. The σ function has a wide range of choices [SD14] such as *logistic function* (*sigmoid*), *hyperbolic tangent function* (*tanh*) [LBOM12], or *rectified linear unit* (ReLU) [HSM+00].

For multiclass classification, a *softmax* or *normalized exponential* operation is applied in the output layer:

$$y_j = softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{2.19}$$

where z_j is the pre-softmax input of the j^{th} neuron. The output activations are squashed within $(0, 1)$, and the sum of them is 1. The softmax output $Y(y_1, y_2, ..., y_N)$ can be regarded as the posterior probabilities conditioned on the input X of the neural network.

When training the network for multiclass classification, normally the categorical *cross entropy* cost function [Mur12] is used:

$$cost = -\sum_{j=1}^{N} g_j \log y_j, \tag{2.20}$$

where g_j is the ground truth probability of the j^{th} class (either 0 or 1), and y_j is the posterior probability of the j^{th} class given the input feature. The network weights are updated through *gradient descent* along the direction that minimizes the cost. Through this process, the error at the output layer will "back-propagate" to the whole network. Hence this method is called gradient descent with *back-propagation* [RHW88].

The most commonly used optimization procedure is called mini-batch *stochastic gradient descent* (SGD). It is an iterative training process. At each iteration a batch of samples are drawn from the training set, and then the gradient descent with back-propagation is performed to update the network's weights once. This process is repeated until the stopping criteria is satisfied.

Deep Belief Network — DBN is a generative model that captures the joint probability distribution of several layers of latent variables and the visible variables that they generate. Figure 2.11 illustrates a DBN with three hidden layers. An extra output layer can be added on top of the last hidden layer with full connections in order to turn it into a discriminative model. Note that the top two hidden layers have undirected connections, while the others have top-down directed connections.

A discriminative DBN is trained in two phases. Firstly, it is *pre-trained* as a generative model using the unlabeled data. This unsupervised training phase aims at reconstructing the input. Then it is *fine-tuned* as a discriminative

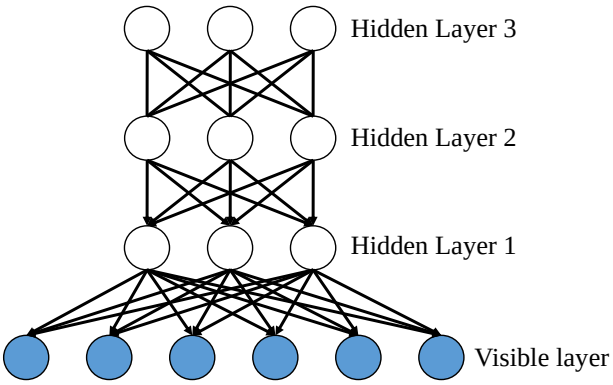

Figure 2.11 A deep belief network with 3 hidden layers

model using the labeled data. This supervised training phase on one hand tries to fine-tune the reconstruction weight, and on the other hand aims at classification.

The generative pre-training process is conducted by training several *restricted Boltzmann machines* (RBM) in series [HOT06]. An RBM [Smo86] is an energy based probabilistic model, visualized as an *undirected bipartite graph*, which is able to learn a joint distribution of all the nodes. A trained RBM can reconstruct its input by sampling from the network. In practice, this property makes it suitable for representation transformation.

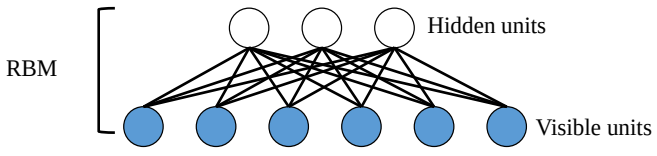

Figure 2.12 Restricted Boltzmann machine

Figure 2.12 shows an RBM. The *energy* of the network is defined as:

$$-E(v,h) = \sum_i a_i v_i + \sum_j b_j h_j + \sum_i \sum_j v_i w_{i,j} h_j, \tag{2.21}$$

where v stands for the visible units, h stands for the hidden units, a and b are visible and hidden bias. The joint probability of v and h can thus be defined as:

$$P(v,h) = \frac{1}{Z} \sum_h e^{-E(v,h)}. \tag{2.22}$$

which is in proportion to the negative energy. In this equation, Z is a *partition function* that sums $e^{-E(v,h)}$ for all possible configurations, so that $P(v,h)$ accumulates to 1. Assume that all visible units are conditionally independent, we have:

$$P(v|h) = \prod_i P(v_i|h). \tag{2.23}$$

Similarly, assume that all hidden units are conditionally independent, there is:

$$P(h|v) = \prod_j P(h_j|v). \tag{2.24}$$

Each unit's activation is normally *stochastic binary* [Hin10], i.e., it takes a binary 1 with a probability of its activation value:

$$\begin{aligned} P(h_j = 1|v) &= \sigma(b_j + \sum_i w_{i,j} v_i) \\ P(v_i = 1|h) &= \sigma(a_i + \sum_j w_{i,j} h_j). \end{aligned} \tag{2.25}$$

The gradients of RBM's reconstruction error with respect to the weights can be easily expressed as:

$$\frac{\partial \log p(v)}{\partial w_{i,j}} = < v_i h_j >_{data} - < v_i h_j >_{model} \tag{2.26}$$

where "angle brackets are used to denote expectations under the distribution specified by the subscript that follows" [Hin10]. But the difficulty of computing

the gradients is that $< v_i h_j >_{model}$ is an inherit value of the model itself, which takes a long time to converge.

A practical way to train an RBM is to use SGD with *contrastive-divergence* [Hin10] (CD) or *persistent-contrastive-divergence* (PCD) [Tie08]. Both methods compute an approximation of $< v_i h_j >_{model}$ by taking only a few steps along the *Markov chain* ($v_0 \rightarrow h_0 \rightarrow v_1 \rightarrow h_1 \rightarrow ... \rightarrow v_t \rightarrow h_t$), without waiting for convergence.

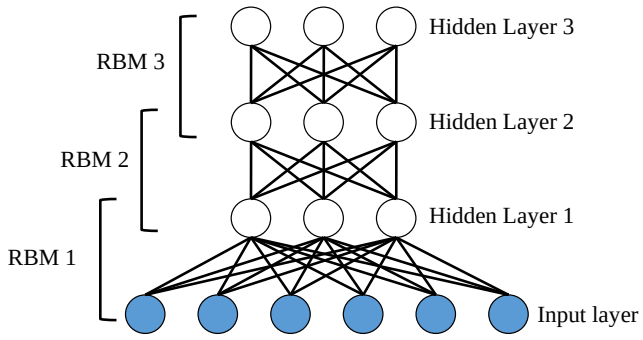

Figure 2.13 Deep belief network pre-training phase

Pre-training the DBN in Figure 2.13 takes the following steps:

1. Regard the network as a stack of RBMs, and order them bottom-up;
2. Train the first RBM;
3. Transform the input bottom-up through all trained RBMs and train the next RBM;
4. Repeat the previous step until there is no RBM left untrained.

The fine-tuning phase is similar to training an FCNN. The network weights are first initialized by the outcome of the pre-training phase. Then all connections

will be turned to feedforward and the cost from the output layer will back-propagate to update the network's weights.

Feature Learning in ACE — Hamel and Eck [HE10] are among the first to introduce feature learning into MIR. Specifically, they train a DBN as a frame-wise music genre classifier, and then take the last hidden layer's activation as the learned feature. Later, Humphrey et al. [HBL13] explore the theoretical underpinnings of feature learning in music informatics. Sigtia and Dixon [SD14] summarize previous works and introduce more feature learning techniques to MIR.

The first feature learning approach in ACE is proposed by Humphrey and Bello [HB12], who use a CNN to extract features. Instead of learning an "intermediate" feature representation, the CNN outputs a "probability surface", which is actually the posterior probabilities of chords conditioned on the CNN's "time-pitch" constant-Q spectra input. In this work, neither the concept of "chroma" nor "pitch class" is involved.

Boulanger et al. [BLBV13] apply a DBN based approach similar to Hamel and Eck's. They first perform a PCA on an STFT spectrogram to reduce input dimensionality, and then they train a DBN to predict the output at each frame. The activations of the last hidden layer are used as the learned features. Again there is neither the concept of "chroma" nor "pitch class" in this work. Later, Zhou and Lerch [ZL15] augment this process with a time-frequency splicing function inserted between the PCA and the DBN in order to exploit similarities of neighboring frames. Along the same line, Sigtia et al. [SBLD15] use an FCNN to classify the chord label at each frame. They also

use the activations of the last hidden layer as the learned features, which will go through a mean pooling process before pattern matching.

Features can not only be learned from chord labels, but also from chord templates. Fillip and Gerhard [KW16a] use binary chord templates as targets to train their feature extractor, i.e. the "deep chroma extractor", which is able to obtain highly clean chromagram from the audio.

2.2.4 Pattern Matching

The extracted or learned features are the input of a pattern matching process, which models the relationship between the features and the chords:

$$chords = M(features), \tag{2.27}$$

where *chords* contain both *labels* and *segmentation*, and M is the pattern matching model.

Template Matching with Post-Filtering — A straightforward way to implement M is by *template matching* with *post-filtering*, which is first proposed by Fujishima [Fuj99]. This method predefines a "chord template" (CT) for each chord. A CT is a 12-dimension binary vector, indicating whether each of the 12 pitch classes presents at a chord. A distance measure, such as Mahalanobis distance[RSN08], Euclidean distance [ZR07], or cosine similarity [HS05], is used to measure the similarity between a PCP and a CT. As a result, for each PCP (or chroma), the chord that "matches" is the one with the highest similarity, or the lowest distance. Ouder et al. [OGF09] augment this method by incorporating an "exponentially decreasing spectral profile" [Góm06b] into CT, and using Kullback-Leibler divergence [KL51] as the similarity measure.

A practical chord rendering will probably scatter the chord tones throughout the whole chord duration, instead of filling everywhere with all chord tones. Therefore, a post-filtering process must be performed to smooth the template matching output in order to capture the chord tone distribution over time. Similar to the pre-filtering [Cho14] in feature extraction, the post-filtering techniques include low-pass filtering [OGF09] and median filtering [HS05, HB12].

Gaussian Mixture with Hidden Markov Model — hidden Markov model (HMM)

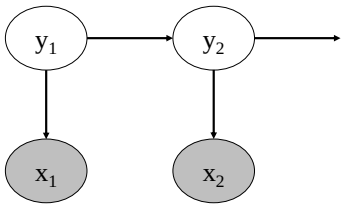

Figure 2.14 Hidden Markov Model

(Figure 2.14) formalizes a practical scenario where an observable sequence $X = (x_1, x_2, ..., x_n)$ is generated by a hidden sequence $Y = (y_1, y_2, ..., y_n)$ that satisfies the *Markov assumption* [G+85], that each state y_n only depends on its immediate predecessor y_{n-1}:

$$P(y_n|y_{n-1}, y_{n-2}, ..., y_1) = P(y_n|y_{n-1}). \tag{2.28}$$

Moreover, HMM is also subjected to the *output independence assumption*, which confines that each observation x_n is dependent only on the hidden state y_n. Particularly, an HMM models the joint probability $P(X, Y)$, which can be factorized as:

$$P(X, Y) = \prod_{n=1}^{N} P(x_n|y_n)P(y_n|y_{n-1}), \tag{2.29}$$

where $P(y_1|y_0) = P(y_1)$. Based on this equation, an HMM can be parameterized by three sets of probabilities:

- the *prior probabilities* $P(y_1)$, which determines the *a priori* of all hidden states;
- the *transition probabilities* $P(y_n|y_{n-1})$, which determines transition weights between any pair of states;
- the *emission probabilities* $P(x_n|y_n)$, which are random distributions that model the generative relationship between the hidden states and the observation.

By the time of the emergence of ACE, HMM had been extensively used in ASR [Rab89, HAJ90] because of its unparalleled sequence modeling power. Borrowing this idea from ASR, Sheh and Ellis [SE03] propose an HMM based sequence decoder for ACE, the core idea of which is to assign the chromagram as the observable variables and the chord labels as the hidden variables, as depicted in Figure 2.15. In their work, each chroma has 24-dimension, and the HMM's emission probability $P(x|y)$ is a Gaussian distribution in 24-dimension. The model parameters are trained using the Baum-Welch algorithm [BPSW70] with unlabeled data. Afterwards, the hidden variables (i.e. chords) are decoded using the Viterbi algorithm [Rab89], which applies dynamic programming to find out the most likely sequence of hidden states conditioned on the observable evidence. This design paradigm of sequence transcription system is widely known as "Gaussian mixture model - hidden Markov model", or GMM-HMM.

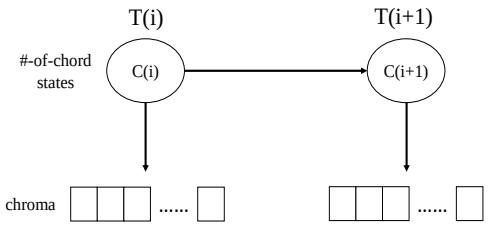

Figure 2.15 A hidden Markov model for chromagram decoding, where "T" stands for "time" and "C" stands for "chord".

GMM is but one type of model that can be coupled with HMM. In fact, any *generative model* that is able to provide the *likelihood* $P(x|y)$ is a suitable candidate (for example, Burgoyne and Saul [BS05] use a *Dirichlet model*). Moreover, given a uniform distributed prior for all hidden states, any *discriminative model* that provides the *posterior probability* $P(y|x)$ is also adoptable. This can be seen via the Bayesian rule:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}, \tag{2.30}$$

or verbally: "posterior is proportional to prior times likelihood". For example, the naive "template matching" method is actually a discriminative approach that outputs the posterior probabilities of chords given the chroma observation. It can also be coupled to an HMM for chord decoding [RK08]. In this case the function of the HMM is smoothing, or post-filtering. By regarding the posterior $P(y|x)$ to be "observable" (since it is easily computable from the chroma), it reduces to a "template matching - dynamic programming" problem without any "hidden variables" involved. Alternative to template matching, such discriminative models can be a neural network [ZG08], a support vector machine [WEJ09] or others that output chord posteriors.

The GMM-HMM model is widely adopted in many ACE systems. The model's parameters can be chosen via expert knowledge [BP05, WD08, CMD+13], or trained via ground truth data [EC07, LS08, WSDR09, KO09, MD08, RUS+09, CB09].

Musical Context Constraints — Both the naive post-filtering and the HMM decoding impose higher level constraints to the raw output. In fact, chords can be correlated with *key* and *beat*. Note that:

- chord progression formed by the diatonic chords, especially the tonic (I), dominant (V), or sub-dominant (IV) chords, strongly indicates the underlying the key;
- chord changes usually happen at the start of a beat, especially on the *downbeat*, which is the first beat of a measure.

In most pop/rock music, the key of a song does not change. In some other cases, the key will *modulate* from time to time, and the modulation usually happens after the key's establishment.

Taking advantages of these musical knowledge, the chord sequence output can be "guided" or "corrected" accordingly. Yoshioka et al. [YKK+04] pioneer the use of both key and beat information in ACE to achieve a "concurrent recognition of chord boundaries, chord symbols and keys" under their "hypothesis-search" algorithm. Around the same time, Maddage et al. [MXKS04] propose another algorithm that estimates chords in the first pass, and then dynamically extracts the key and beat information to correct chord annotations and boundaries. With a similar notion, Shenoy and Wang [SW05] leverage key and beat information to perform two phases of "chord accuracy enhancement"

following a certain set of predefined rules derived from the musical knowledge. Many other ACE approaches [LS08, ZR07, SIY+08, RSN08, KO11] follow this musical context assisting trend.

A unified probabilistic chord-key framework is first proposed by Catteau et al. [CML07]. A unified chord-beat model is first proposed by Papadopoulos and Peeters [PP08]. Other variants are put forward by Pauwels and Martens [PM10, PM14], and Weil and Durrieu [WSDR09]. Noland and Sandler provide a detailed study of chord-key integrated models [NS09]. These models are all extended from the classic GMM-HMM, and all of them can be solved by dynamic programming based algorithms. These lead to a more unified musical probabilistic framework, which will be elaborated in the following.

Dynamic Bayesian Network — A *Bayesian network* [Pea14] is a probabilistic model of random variables and their conditional dependencies through a *directed acyclic graph* (DAG). A *dynamic Bayesian network* [Mur02] (DYBN, not to be confused with the DBN mentioned above) is a Bayesian network in which the random variables can also relate to each other over time. It is a generalization of HMM, which can only model one hidden variable and one observable variable over time. DYBN allows multiple hidden and observable variables with more flexible dependencies.

Mauch and Dixon [MD10a] pioneer the integrated model of musical context as illustrated in Figure 2.16. This DYBN model (GMM-DYBN) integrates the probabilistic models of beat (metric position), key, chord and bass, and it is parametrized by the expert knowledge on the interrelationships among these musical elements. Later, Ni et al. [NMSRDB12] and McVicar [McV13] follow up this work with data driven DYBN parameterizations.

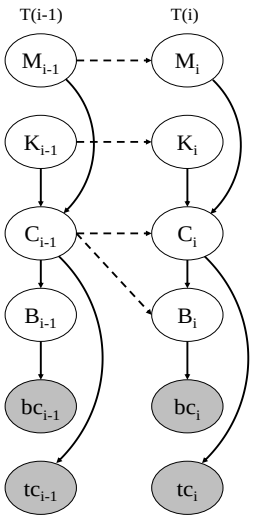

Figure 2.16 A dynamic Bayesian model for automatic chord estimation. M: metric position; K: key; C: chord; B: bass; bc: bass chromagram; tc: treble chromagram

Conditional Random Fields and Support Vector Machines — An HMM (and also its general case DYBN) can be regarded as a generative model that is either trained or manually engineered to maximize the joint probability of $P(X, Y)$, where X is the observable sequence, and Y is the hidden sequence. Note that $P(X, Y) = P(Y|X)P(X)$, the model does not only look for discriminative rules that predict Y from X, but also the distribution of X. As pointed out by Sutton et al. [SMR07], it could be a waste of modeling power "when the task does not require being able to generate X, such as in segmenting and labeling", which is exactly the case in ACE. Therefore, a good alternative for the task is a pure discriminative model, which only models $P(Y|X)$.

In view of this, Weller et al. [WEJ09] implements a solely discriminative system with a *support vector machine* (SVM) [CV95] based on the SVMstruct algorithm [TJHA05]. Burgoyne et al. propose a system [BPKF07] that incorporates a linear-chain *conditional random field* (CRF) [LMP01], where each

hidden state depends on the complete observable sequence. More generally, Papadopoulos and Tzanetakis [PT12] use the *Markov logic network* (MLN) [RD06] to take advantages of both *first-order logic* and probabilistic graphical models. A general comparison of different sequence labeling algorithms can be found in [NG07].

Further Readings — There are a few review articles on ACE pattern matching. Papadopoulos and Peeters [PP07] review both the expert knowledge and machine learning systems with template matching, GMM-HMM, or musical context constrained techniques under the same large-scale benchmark. McVicar et al. [MSRNDB14] compare an extensive list of seven types of models, including the high-order HMMs [SVB09, MDH$^+$07, KO09, YG11], and genre-specific models [LS08, Lee07].

2.2.5 Deep Learning for Pattern Matching

It can be expected that deep learning is not only suitable for feature learning, but also pattern matching. Here we introduce the mechanism of the *recurrent neural network* (RNN) and its *long short-term memory* (LSTM) units, as they are quite frequently used in ACE for this purpose.

Recurrent Neural Network — RNN is a neural network with cyclical connections, so that the network can be recurrently expanded into multiple frames [Elm90, Jor86, LWH90]. Figure 2.17 shows a simple RNN with one self-connected hidden layer, unfolded into four frames. This network has three sets of weights, namely, A, B and R, which are referred to as "input matrix", "output matrix" and "recurrent matrix" respectively. At time t, given the input frame X^t and the previous hidden layer activation H^{t-1}, the hidden activation

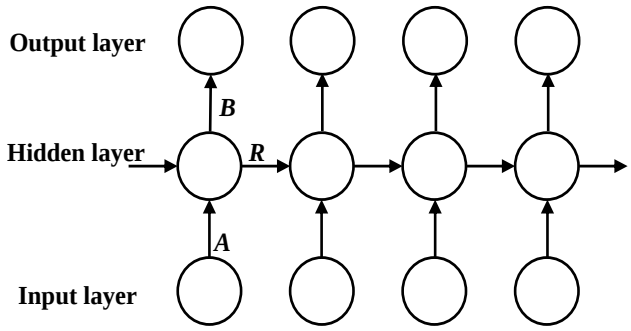

Figure 2.17 Recurrent neural network

will be:

$$H^t = \sigma(\sum_j A_{i,j} X_j + \sum_{i'} R_{i,i'} H_{i'}^{t-1}), \tag{2.31}$$

where R is a square matrix, and σ is a non-linear activation function. The network's output Y^t is:

$$Y^t = softmax(\sum_i B_{i,k} H_i^t). \tag{2.32}$$

RNN is a discriminative model that captures the conditional probability of the output sequence $Y(Y^1, Y^2, ...)$ given the input sequence $X(X^1, X^2, ...)$:

$$P(Y|X) = RNN(X). \tag{2.33}$$

RNN models this probability in a sequential manner, so that Y^t is conditioned on all previous inputs $X^{1:t}$. If a backward hidden layer is added to the model, Y^t will be conditioned on the whole input sequence X. This modified model is called *bidirectional recurrent neural network* (BRNN), as illustrated in Figure 2.18.

To train the network, a categorical cross entropy loss function similar to equation 2.20 is commonly used. The gradients are computed as a backward pass through the network from each output node. This leads to the

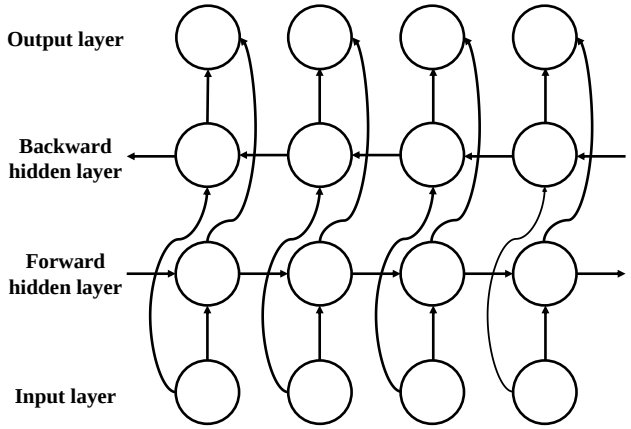

Figure 2.18 Bidirectional recurrent neural network

back-propagation-through-time (BPTT) [RMG+88, Wer90] technique. However, when the training sequence is long, the gradient signal may die down gradually through back-propagation. This *gradient vanishing* [Ben09] phenomenon often makes the training ineffective or unsuccessful. Using LSTM units [HS97] instead of normal neurons is a common way to circumvent this undesirable effect.

Long short-term memory — Figure 2.19 shows the structure of an LSTM unit

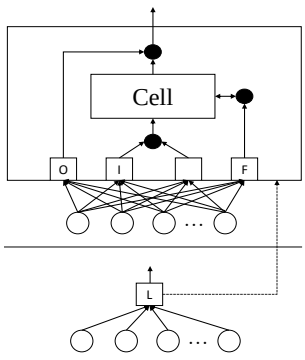

Figure 2.19 Long short-term memory

(indicated as "L") with a typical configuration [Gra12]. The input data path

has 4 identical copies for input gate I, output gate O, forget gate F and the input port (no letter). Each gate or port will activate an output between 0 and 1 according to its inputs and activation function. The input gate activation is multiplied with the input port activation to become an input value to the LSTM cell. The forget gate activation is multiplied with the cell value from the previous time step to become another input to the cell. The current cell value is determined by the sum of the above two cell inputs. The output of the LSTM unit is given by the multiplication of the output gate activation and the current cell value. Note that in some configuration the cell value can also be fed back into the gates.

Now let's assume the RNN in Figure 2.17 is implemented with LSTM hidden layer. The original input matrix A will have four different instances A_i, A_f, A_o and A_c, for input gate, forget gate, output gate and input port respectively. Similarly, the recurrent matrix R will also have four different instances R_i, R_f, R_o and R_c. Let b be the bias input. Formally, the input gate activation at time t is:

$$i_t = \sigma(A_i X^t + R_i H^{t-1} + b_i), \tag{2.34}$$

the forget gate activation is:

$$f_t = \sigma(A_f X^t + R_f H^{t-1} + b_f), \tag{2.35}$$

the input port activation is:

$$C_0^t = \sigma(A_c X^t + R_c H^{t-1} + b_c), \tag{2.36}$$

the current cell value is:

$$C^t = i_t * C_0^t + f_t * C^{t-1}, \tag{2.37}$$

the output gate activation is:

$$o_t = \sigma(A_o X^t + R_o H^{t-1} + b_o), \tag{2.38}$$

and finally the output of the LSTM unit is:

$$H^t = o_t * \sigma(C^t). \tag{2.39}$$

where $*$ stands for element-wise multiplication.

Deep pattern matching in ACE — Here we define a few terms that help place each of the following approaches in a tentative ACE taxonomy:

- **local** means the process sees a window of frames as the input context;
- **global** means the process runs with awareness of the full input;
- **smoothing** means a segmentation process that aims at post-processing or post-filtering the raw classification results.

By these definitions, we could place the traditional ACE approach pioneered by Fujishima [Fuj99] and Sheh and Ellis [SE03] as the "local feature extraction - global smoothing" approach.

We now introduce several deep learning techniques for ACE pattern matching. Since this is highly related to the "feature learning", we will refer back to the feature learning techniques when necessary.

The first deep learning based pattern matching in ACE is also proposed by Humphrey and Bello [HB12]. They implement a CNN to classify chords at each constant-Q frame, and then apply median-filtering to smooth the classification results into segments. The CNN outputs a "probability surface", which is the posterior chord probabilities conditioned on the input frame, so that it actually

performs both feature learning and pattern matching. Consider that the role of CNN here can be replaced by any type of deep neural nets, this is a "local classification - global smoothing" approach.

Boulanger et al. [BLBV13] perform feature learning via an STFT-PCA-DBN chain, and take the activations of the last DBN hidden layer as the features. The features are then classified by an RNN, whose output is further smoothed using an HMM with Viterbi decoding, or beam search with dynamic programming. Sigtia et al. [SBLD15] follow up this work by applying the LSTM-RNN, and augmenting the beam search with a hashed beam search. These can be regarded as the "local feature learning - global classification - global smoothing" approaches.

Zhou and Lerch [ZL15] put forward another system that resembles Boulanger's in terms of the feature learning. They use an SVM to classify the learned features locally, and use an HMM to perform sequence decoding and smoothing. This belongs to the "local feature learning - local classification - global smoothing" approach.

Therefore, according to our taxonomy, by far there are four types of ACE approaches:

- local feature extraction - global smoothing [Fuj99, SE03]
- local classification - global smoothing [HB12];
- local feature learning - global classification - global smoothing [BLBV13, SBLD15];
- local feature learning - local classification - global smoothing [ZL15].

We will refer back to this taxonomy in Chapter 3 and 4 when we introduce our new approaches.

2.2.6 Chord Vocabulary and Output Format

Chord vocabulary is a big issue in ACE as it defines the problem domain. The size of a chord vocabulary is denoted by the number of chord classes, which equals to twelve time the number of chord types (a chord type can be rooted at any of the 12 pitch classes in 12-tone equal temperament). In the two pioneering ACE works, Fujishima's system [Fuj99] supports 324 classes, including many chord types used in jazz; and Sheh and Ellis's system [SE03] has 84 classes with 7 types, including maj, min, $maj7$, $min7$, 7, aug and dim.

Consideration for Small Vocabulary — Note that a large vocabulary may lead to poor system performance in practice. One reason is that in the early days of ACE there were not enough amount of ground truth data to train a large vocabulary system, particularly for the "long-tail" chords. For example, Burgoyne et al. [BWF11] report that in Western pop song practice, maj, min, 7, $min7$ and $maj7$ make up of more than 80% of the population. These chords are considered "ordinary" or "common", while the others are considered "long-tail", or "uncommon".

On the other hand, for an expert knowledge driven system, it is also hard to recognize long-tail chords since most of them are slightly *extended*, *altered* or *suspended* upon the ordinary chords. Their sound qualities can be distinguished by human though, their underlying note compositions differ in a very subtle way. If an ACE *expert system* tries to capture the common chords as much as possible, it finds difficult to secure the uncommon chords in the same

way. Moreover, since the population of uncommon chords is small, even a few misclassification may lead to severe performance drop in those categories.

In this case, if a system tries to support a large vocabulary, it may suffer from low overall chord symbol recall due to confusions between the common and uncommon chords. Specifically, due to the extremely unbalanced population, there will be many common chords misclassified as uncommon, and only a few the other way around. However, if a system only supports common chords (i.e. a small vocabulary), although all the uncommon chords will be misclassified, it guarantees the performance of the majority.

Su and Jeng [SJ01], Maddage et al. [MXKS04] and Yoshioka et al. [YKK+04] are among the first to use a 48 chords vocabulary with four types, including *maj*, *min*, *aug* and *dim*. This is echoed in many later works [HS05, CML07, BPKF07, SJ01, PP08]. Bello and Pickens [BP05] propose an even smaller vocabulary, covering only 24 chords with *maj* and *min* (normally called *majmin*). This is also adopted in many systems [RK08, WD08, KO09, WEJ09, NMSRDB12, CWB10, HB12] and thus *majmin* becomes the mainstream in ACE. Note that usually a "no chord" with a "chord symbol" N or $N.C.$ will also be included in the vocabulary to denote everything that is "not a chord", such as silence, speech, natural soundscape or environmental noise.

Consideration for Large Vocabulary — Large vocabulary systems have come to resurgence recently. This may be partly due to the growing complexity of the graphical probabilistic models, and partly due to the use of more complicated machine learning techniques together with much more ground truth annotation data than two decades ago.

Besides the vocabularies in two early systems [Fuj99, SE03], there are currently several other types of large vocabularies being used. Mauch [Mau10] puts forward a vocabulary in his DYBN based system, which he names as "full". It contains *maj*, *min*, *maj*/3, *maj*/5, *maj*6, 7, *maj*7, *min*7, *dim*, *aug*, totally 121 chord types (later on this vocabulary is refereed to as Full121). Note that this is a vocabulary with chord inversions *maj*/3 and *maj*/5. This vocabulary is adopted in several systems [NMSRDB12, MSRNDB14, BLBV13].

Cho [Cho14] proposes two vocabularies with 61 chords [4] and 157 chords [5] respectively, without inversions [BdHP14]. They are also used in Humphrey's systems [HB15, Hum15].

Pauwel and Peeters [PP13], in an effort to introduce a new set of ACE evaluation methods (will be discussed in Section 2.3), bring in a new vocabulary called *SeventhsInv* or *SeventhsBass* that contains *maj*, *min*, *maj*7, *min*7, 7 with all of their inversions [6], and the "N.C" [7].

Notably, Mauch's popular open source ACE tool, Chordino [CLS10], which features NNLS bass-treble chroma with HMM decoding instead of DYBN, features a default vocabulary of 181 chords with some inversions [8].

Consideration for Chord Inversions — Thus far, there are only a few ACE systems that supports inversions [CLS10, Mau10, NMSRDB12, McV13]. When a system supports inversions, it's overall performance could be seriously downgraded. This is because inversions are easily confused with their root positions,

[4] *maj*, *min*, *maj*7, *min*7, 7

[5] *maj*, *min*, *maj*7, *min*7, 7, *maj*6, *min*6, *dim*, *aug*, *sus*4, *sus*2, *hdim*7 and *dim*7

[6] *maj*, *min*, *maj*7, *min*7, 7, *maj*/3, *min*/*b*3, *maj*7/3, *min*7/*b*3, 7/3, *maj*/5, *min*/5, *maj*7/5, *min*7/5, 7/5, *maj*7/7, *min*7/*b*7, 7/*b*7

[7] means not a chord

[8] *maj*, *min*, *maj*/3, *maj*/2, *maj*/5, *dim*, *aug*, *maj*6, *min*6, 7, *maj*7, *min*7, *dim*7, 7/3, 7/*b*7

whose population is much larger. Ignoring inversions makes such confusion only possible in one direction (inversions misclassified as root position only), but supporting inversions makes it possible in both directions.

The sound quality of inversions and root positions are totally different in many musical contexts. For example, in Figure 2.20, inversions are introduced to keep bass continuities or to indicate a shift of key. If an ACE system does not support inversions, it definitely could break bass line continuations and, thus, changes the original harmonies.

1) | G | D/F# | F | C/E | Cm/Eb |
2) | A | Bm | A/C# | D |
3) | C | G/B | Am | Am/G | F | C/E |
4) | C | F | C/E | D/F# | E/G# | F#/A# | Bm7 | C# |

Figure 2.20 Four chord progressions that contain bass line continuations which demand chord inversions. Progressions like 1, 2 and 3 are very popular among pop/rock. Progression 4 induces a key shift from C major to F# minor.

The Output Format — The output format of ACE is defined by Harte et al. [HSAG05]. The following is the ground truth annotation of the first few seconds of *Let it be*:

0.00 0.18 N

0.18 1.85 C

1.85 3.45 G

3.45 4.72 A:min

4.72 5.12 A:min7/b7

5.12 5.95 F:maj7

5.95 6.77 F:maj6

A chord label is preceded by two time stamps, being the onset and offset time of the chord in terms of seconds. These time stamps are actually the chord "segmentation". As for the label, a chord's root is first notated, followed by a semicolon ":" to separate the root and the quality. An inversion is notated by a slash "/", followed by the bass. If a chord is maj, it can be simply notated as the root only.

2.3 Review of ACE Evaluation

This section is going to review several issues regarding the evaluation of ACE systems. First of all the system performance metrics will be discussed. Then under the context of a annual music information retrieval exchange event, the actual application of the metrics to different chord matching functions will be elaborated. The section is concluded with a fundamental issue that is closely related to evaluation - human annotation subjectivity.

2.3.1 Estimation Accuracy Metrics

In his pioneering ACE paper, Fujishima [Fuj99] scores the system with:

$$score = \frac{\#\ correct\ guesses}{\#\ all\ guesses}, \tag{2.40}$$

where a *guess* is made on every frame. This is later referred to as *frame-based chord symbol recall* [Har10], *average overlap score* [Oud10], or *relative correct overlap* (RCO) [Mau10]. Generally, these scores can all be regarded as *chord symbol recall* (CSR), formulated as:

$$CSR = \frac{|\text{correctly identified frames}|}{|\text{total number of frames}|} \times 100\%. \tag{2.41}$$

It is better to clarify why this can be a *recall*. As we know, "recall" can be expressed as:

$$recall = \frac{TP}{TP + FN}, \tag{2.42}$$

where TP are true positives, and FN are false negatives. For example, regarding the $Cmaj$ chord, TP stands for the portion where the $Cmaj$ classifications match the $Cmaj$ ground truths, and FN is the portion where the ground truths are $Cmaj$s but the classifications do not match them (see Section 2.3.2 for more about "match"). Thus the recall for $Cmaj$ can be written as:

$$recall_C = \frac{TP_C}{TP_C + FN_C}. \tag{2.43}$$

Note that a multiclass classification problem can be reduced to a one-v.s.-all classification problems when looking at just one specific class such as $Cmaj$. And since the total number of $Cmaj$ chords in this piece is $N_C = TP_C + FN_C$,

$$recall_C = \frac{TP_C}{N_C}. \tag{2.44}$$

The same is also true for the other chords, such as Am:

$$recall_{Am} = \frac{TP_{Am}}{N_{Am}}. \tag{2.45}$$

Assuming there are only these two chords in the dataset, the overall recall is the weighted sum of the two recalls:

$$recall = \frac{N_C * recall_C + N_{Am} * recall_{Am}}{N_C + N_{Am}} = \frac{TP_C + TP_{Am}}{\#\,of\ frames}, \tag{2.46}$$

which is equivalent to Equation 2.40 and 2.41. Strictly speaking this is better termed as the *accuracy* metric, which is defined exactly as the correctly classified samples divided by the total number of samples. It is worth noting that the above "weighted sum" process in deriving the overall recall is actually

a "bias", that classes with larger portions are more important than those with smaller portions.

Instead of computing the *CSR* in a frame-based way, Harte [Har10] introduces a *segment-based chord symbol recall*:

$$CSR = \frac{|S \cap S^*|}{|S^*|} \times 100\%, \tag{2.47}$$

where S and S^* represents the automatic estimated segments, and ground truth annotated segments, respectively, and the intersection of S and S^* results in the parts where they overlap and have equal chord annotations. Verbally, Equation 2.47 can be re-expressed as [9]:

$$CSR = \frac{\text{total duration of segments where chord annotation equals chord estimation}}{\text{the total duration of annotated segments}} \times 100\%, \tag{2.48}$$

For multiple tracks estimation, the overall performance can be reported as the *weighted chord symbol recall* (WCSR):

$$WCSR = \frac{\sum_i Length(Track_i) * CSR_i}{\sum_i Length(Track_i)} \times 100\%, \tag{2.49}$$

where subscript i denotes the track number. It is the weighted average of all tracks' *CSR*s by the lengths of these tracks. This is equivalent to the *total relative correct overlap* (*TRCO*) used by McVicar [McV13].

ACE researchers also report *WCSR* scores on individual chords [Mau10], as well as the confusion tables among chords [Mau10, Oud10, Pap10, Kha11]. The *WCSR* of a specific chord type is:

$$WCSR_C = \frac{\sum Length(C_i) * CSR_i}{\sum Length(C_i)}, \tag{2.50}$$

[9] `http://www.music-ir.org/mirex/wiki/2015:Audio_Chord_Estimation`

where the subscript i denotes the i^{th} instance of chord C within the data set. A different view towards the overall performance is to take the average *WCSR*s of all chords without any weighting. This is called the *Average Chord Quality Accuracy* (*ACQA*) [Cho14], which could measure the well-roundedness or balanced performance of a model:

$$ACQA = \frac{\sum WCSR_C}{\#\ of\ chords}. \tag{2.51}$$

Models that over-fit on a few chord types tend to get lower *ACQA*s, while those well balanced ones will have higher *ACQA*s.

2.3.2 Chord Matching Functions

All the above metrics compute scores only if a *chord matching function* is defined so that the evaluation algorithm knows what is a "match". Such function takes the form of [Har10]:

$$M(C_G, C_E) = \begin{cases} 1 & \text{if } C_G \text{ matches } C_E \\ 0 & \text{otherwise} \end{cases} \tag{2.52}$$

where C_G is the ground truth label and C_E is the estimated label. At the early years of ACE, chords are matched as what they are [Fuj99, SE03, BP05]: two labels are the same if and only if they are the same label. Since then, different matching functions have been used in MIREX (Music Information Retrieval Evaluation Exchange) [10] [Dow08], which is a annual event that aims at advancing various MIR technologies through extensive open evaluations. ACE has been one of the MIREX tasks since 2008. Because different systems may support different vocabularies, various matching functions have been proposed to try to make fair comparisons among them.

[10] `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

MIREX ACE 2008 and 2009 target the evaluation on *majmin* vocabulary. A pre-processing stage maps all chords to *majmin* following the predefined rules set up by the MIREX officials [Har10]. The most common "rules" are to map all major type chords (e.g. 7, 9, *maj*7, *maj*9, etc.) to *maj*, and all minor type chords to *min*, but disagreements may occur regarding chords such as *aug*, *dim*, *sus*4 and *sus*2.

From 2010 to 2012, MIREX ACE uses a different matching function. It is a "bag of chroma" matching strategy, as discussed by Pauwel and Peeters [PP13], that a chord is first expanded as a binary pitch class chroma, and two chords match each other if their chroma intersection has 3 or more bins. This scheme is criticized by Pauwel and Peeters [PP13] as:

- not able to distinguish chords with same chroma but different roots;
- not able to penalize a chord with superfluous chroma

Both the *majmin* mapping and "bag of chroma" chord matching functions are essentially stemming from a fact: while the ground truth has a vocabulary of unlimited (or very large) size, the ACE output has not. This contradiction takes effect when the chord estimations are to be compared against the manual annotations. Mechanisms must be invented to make sure that they are reasonably comparable in any case. In this sense, unless an ACE system can nicely support an unlimited vocabulary, any chord matching function will be an approximation. *majmin* mapping simplifies everything to be *maj* and *min*, neglecting all subtle different flavors of chords, and the results will inevitably favor those systems who only generate *maj* and *min* triads; "bag of chroma",

to the other extreme, tries to avoid the above mentioned contradiction altogether, but actually disregards the importance of chord voicing [11], chord naming, and the containing relationships among chords, thus the results will bias towards systems that pay little attention to these issues. Nevertheless, researchers continue to look for better evaluation methods, or approximations, which is consented [12] to be an essential key to better ACE systems.

In 2013, MIREX ACE adopted a new evaluation scheme [13] [PP13]. The major difference of this method is that it computes the *WCSR* of four different vocabularies: Major and minor (*MajMin*); Seventh chords (*Sevenths*); Major and minor with inversions (*MajMinInv* or *MajMinBass*); and Seventh chords with inversions (*SeventhsInv* or *SeventhsBass*).

It should be clarified that the scores of *MajMin*, *MajMinBass* and *Sevenths* are calculated based on the *SeventhsBass*'s score with toleration of two types of chord confusions:

- Bass confusion: the estimation has the same root and quality as the reference, but has a different bass (e.g., confusion between root position and inversion);

- Seventh confusion: the estimation has the same root and bass as the reference, but has a different quality with regard to the seventh type (e.g., confusion between *maj* and *maj7*, between *7* and *maj7*, or between *min/b3* and *min7/b3*).

With these, we can interpret the scores as follows:

[11] the way a chord is organized vertically (along pitch dimension)

[12] `http://www.music-ir.org/mirex/wiki/The_Utrecht_Agreement_on_Chord_Evaluation`

[13] `https://github.com/jpauwels/MusOOEvaluator`

- SeventhsBass tolerates no confusion;
- Sevenths tolerates all bass confusions;
- MajMinBass tolerates all seventh confusion; and
- MajMin tolerates both.

The rationale behind the toleration is that if such confusion happens, the difference of the original harmony and the modified harmony may still be within an acceptable range. Another perspective to understand this is that sometimes human annotators will also make these confusions. By clarifying that, SeventhsBass is the strictest metric among all; Sevenths and MajMinBass are less strict, both with relaxation of one confusion type; MajMin is the least strict, with both confusions tolerated.

Here are more **original quotes** [14] regarding these chord vocabularies and the evaluation scheme, so that the readers can get a complete idea of what it is:

> 1. With the exception of no-chords, calculating the vocabulary mapping involves examining the root note, the bass note, and the relative interval structure of the chord labels; 2. A mapping exists if both the root notes and bass notes match, and the structure of the output label is the largest possible subset of the input label given the vocabulary; 3. For instance, in the major and minor case, $G : 7(\#9)$ is mapped to $G : maj$ because the interval set of $G : maj$, 1,3,5, is a subset of the interval set of the $G : 7(\#9)$,

[14] http://www.music-ir.org/mirex/wiki/2013:Audio_Chord_Estimation_Results_MIREX_2009

> 1,3,5,b7,#9. In the seventh-chord case, $G:7(\#9)$is mapped to $G:7$ instead because the interval set of $G:7$ 1, 3, 5, b7 is also a subset of $G:7(\#9)$ but is larger than $G:maj$; 4. (As for evaluation of segmentation) we propose to include the directional Hamming distance in the evaluation. The directional Hamming distance is calculated by finding for each annotated segment the maximally overlapping segment in the other annotation, and then summing the differences [ANS+05, Mau10]; 5. Our recommendations are motivated by the frequencies of chord qualities in the Billboard corpus of American popular music [BWF11]

Despite the chord mappings, in many ways this is a much better approximation to an ideal evaluation method. It provides more perspectives into the system performance by considering a spectrum of different chord matching scenarios. Moreover, it is the first official evaluation scheme to incorporate segmentation quality and chord inversions. The segmentation quality (SQ), according to the quote above, is computed as a *directional Hamming distance* (DHD). Assuming S^* and S are the segmentations of the ground truth and the estimation respectively, the DHD from S^* to S is:

$$h(S^*||S) = \sum_{i=1}^{N_{S^*}}(|S_i| - \max_j |S_i^* \cap S_j|), \tag{2.53}$$

where the subscript i indicates the i^{th} segment. Note that the distance is not commutable, which means $h(S^*||S)$ and $h(S||S^*)$ represent two different distances. Conventionally, $h(S^*||S)$ measures the *under-segmentation* and $h(S||S^*)$ measures the *over-segmentation*. In either case, a better segmentation is indicated by a smaller value. When reported as scores, they are usually normalized by the lengths of the tracks, and subtracted by 1, in order to make

it within $[0, 1]$, similar to the *WCSR* score. Harte [Har10] suggests to report one minus the maximum of the two normalized scores, yielding:

$$h(S^*, S) = 1 - \frac{1}{T} \max\{h(S^*||S), h(S||S^*)\} \in [0, 1], \tag{2.54}$$

where T is the total duration of the input. This is officially taken as the MIREX ACE segmentation quality score.

2.3.3 Human Annotation Subjectivity

An even more fundamental problem in ACE evaluation would be: "is the ground truth annotation a gold standard?" Unfortunately, both Ni et al. [NMSRDB13] and Humphrey and Bello [HB15] point out there exists a certain degree of disagreement between two musicians annotating the same dataset, as the latter article concludes that "the subjective nature of chord perception may render objective ground truth and evaluation untenable."

Inevitably, due to differences in musical training, human annotators sometimes disagree, especially on uncommon or long-tail chords [HB15]. In a very strict sense, there is not any "gold standard" if annotators might disagree with each other. But in a loose sense, theoretically there could be a "gold standard" if:

- $G1$: all annotations are done by only one annotator, or
- $G2$: all annotations are done by multiple annotators (much more than two).

In the former case $G1$, the only annotator "dictates" a local "gold standard", so that whenever a machine tries to learn from the data, it actually learns this annotator's "style". In the latter case $G2$, multiple annotators collectively decide

a global "gold standard" in a way such as majority vote, consensus approach [NMSRDB13] or data fusion [KdHV15, Kle04], so that a trained model will aim at the optimal "style" that minimizes the objections among these annotators. Therefore, although the "gold standard" is indeed an important issue, we still have to design a system that "learns well".

In view of all these issues [HB15, NMSRDB13], given a test set that is somewhere between $G1$ and $G2$, perhaps the most "objective" evaluation method is by "subjective" evaluation [Har10]. It requires each piece of chord estimation to be examined by multiple human annotators, as has been correctly pointed out by Humphrey and Bello:

> ...qualitative evaluation should play a larger role in the assessment of automatic systems intended for userfacing applications. If nothing else, users studies can help identify objective measures that align well with subjective experience.

2.4 Research Gaps

This chapter gives a thorough review of ACE, including the musical fundamentals, system design approaches and evaluation methodologies. Through the literature review, we see at least three research gaps.

Firstly, LVACE system is generally overlooked. The concern about disadvantages of LVACE is explained in Section 2.2.6. Moreover, the subjectivity issue in Section 2.3.3 also contributes to the negative concern. Nevertheless, the counterarguments against both concerns are also clear that:

1. to date there are much more training data on uncommon and long-tail chords;

2. parallel to the subjectivity issue, it is of equal importance to design a system that learns well.

Eventually ACE systems should perform comparably with human musicians, hence there is no reason to stop pursuing large vocabulary systems. In addition, it is fully reasonable to design a system to support exactly the vocabulary that is used in the standard evaluation method. Currently the vocabulary used in MIREX ACE is the *SeventhsBass*. But unfortunately, except for Chordino [CLS10], there is no system that handles large vocabulary with inversions (most MIREX submissions support $majmin$, and some with larger vocabularies without inversions).

Secondly, balanced performance on all chord categories is overlooked. It is no need to mention that systems with small vocabulary, such as $majmin$, do not have a balanced performance over a practical set of vocabulary. Even a system that supports a large vocabulary can also over-fit on ordinary chords. Under imbalanced datasets, it is necessary to design proper schemes to keep performances on different classes as balanced as possible.

Thirdly, the chord segmentation information is under-exploited. Evidences have shown that the recent ACE systems have similar segmentation quality performances [BdHP14], while they all perform segmentations and classifications in one single pass. The fact that chords are rhythmically segmented is a fundamental difference between ACE and ASR. Such property should be utilized to build better ACE systems.

This thesis tries to fill in these research gaps with a few novel ideas on LVACE systems.

Chapter 3

A Hybrid GMM-HMM and Deep Neural Nets Approach

Thus far, we have identified several research gaps. Firstly, the existing works have not considered a fundamental difference between ASR and ACE in regards to segmentation, which may lead to a possible design that considers segmentation and classification as two separate tasks. Secondly, the support for large vocabulary has been largely overlooked, particularly the support for chord inversions. Note that chord inversion is a crucial ingredient for pop and rock music, which is the primary focus of ACE research.

This chapter introduces an LVACE system framework that leverages the rhythmic property of chords. It can be categorized as a **"local feature extraction - global segmentation - local classification"** approach, which isolates the process of segmentation from classification. Specifically, the classification process is implemented via deep neural nets. We would like to verify whether this approach could benefit the LVACE performance, in particular the performance on the *SeventhsBass* vocabulary.

Experimental results show that among the three proposed deep neural nets and a baseline model, the RNN based system has the best average chord

quality accuracy that significantly outperforms the other considered models. Furthermore, our bias-variance analysis has identified a glass ceiling as a potential hindrance to future improvements of large vocabulary automatic chord estimation systems.

3.1 System Framework

Figure 3.1 depicts the LVACE system framework in our study. The workflow is as follows:

- Feature extraction: both training and validation data share the same feature extraction module; features are extracted from each input track using a method similar to the one employed in the Chordino system [Mau10] (to be elaborated in Section 3.1.1).

- Segmentation: 1. for training, the feature sequence is segmented by the ground truth annotations; 2. for validation, the feature sequence is segmented using a GMM-HMM process (to be discussed in Section 3.1.2).

- Segment tiling: each feature segment is tiled into a fixed number of sub-segments (see Section 3.1.3).

- Deep neural nets: 1. for training, the segments and their chord labels are used to train the deep neural nets (will be described in Section 3.1.4); 2. for validation, the trained neural network is used to predict chord labels.

3.1.1 Feature Extraction

Feature extraction starts by resampling the raw audio input at 11025 Hz, which is followed by an STFT (with 4096-point Hamming window, 512-point

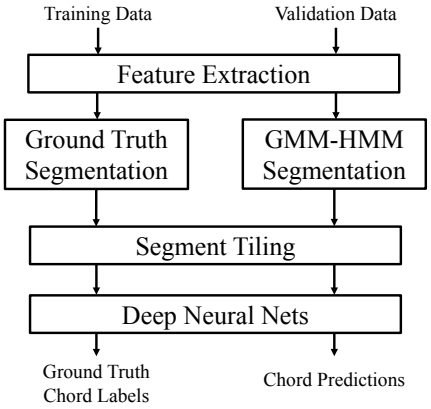

Figure 3.1 GMM-HMM ACE System framework.

hop size). It then proceeds to transform the linear-frequency spectrogram (2049-bin) to log-frequency spectrogram (252-bin, three bins per semitone ranging from MIDI note 21 to 104) using two cosine interpolation kernels [Mau10]. The output at this step is a log-frequency-spectrogram, or log-spectrogram, $Y_{k,m}$, where k is the index of frequency bins, and m is the index of time frames. We denote the total number of frames as M, and the total number of bins in each spectrum as K (in this context $K = 252$).

The amount of deviation from standard tuning is estimated using the algorithm in [DS07], where the amount of detuning is estimated as:

$$\delta = \frac{wrap(-\varphi - 2\pi/3)}{2\pi}, \tag{3.1}$$

where $wrap$ is a function wrapping its input to $[-\pi, \pi)$. φ is the phase angle at $2\pi/3$ of the discrete-Fourier-transform (DFT) of $\sum_m Y_{k,m}/M$. The tuning frequency τ is then computed as:

$$\tau = 440 \cdot 2^{\delta/12}, \tag{3.2}$$

and the original tuning is updated by interpolating the original spectrogram $Y_{k,\cdot}$ at $Y_{k+p,\cdot}$, where:

$$p = (\log(\tau/440)/\log(2)) \times 36. \tag{3.3}$$

The "36" indicates that there are 36 bins per octave (3 bins per semitone) in $Y_{k,\cdot}$. The updated $Y_{k,m}$ spectrogram will be referred to as "notegram" in the following.

To enhance harmonic content and attenuate background noise, a standardization process is performed along the frequency axis:

$$Y_{k,\cdot}^{STD} = \begin{cases} \frac{Y_{k,\cdot} - \mu_{k,\cdot}}{\sigma_{k,\cdot}}, & \text{if } Y_{k,\cdot} > \mu_{k,\cdot} \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

where $\mu_{k,\cdot}$ and $\sigma_{k,\cdot}$ are the mean and standard deviation of a half-octave window centered at $Y_{k,\cdot}$, respectively. This is followed by an NNLS method (equation 2.13) to extract note activation patterns (84-bin, 1 bin per semitone) [MD10a].

The feature dimension is further reduced before the segmentation process. Particularly, each NNLS chroma is weighted by the bass and treble profiles depicted in Figure 3.2. After that the saliences of each pitch class are added together, resulting in a 24-bin bass-treble chromagram. Each column of the bass-treble chromagram is then L_∞ normalized, so that each bin of each chroma is within the range of [0,1]. Table 3.1 provides a summary of different levels of representations generated by this feature extraction process.

We mainly make use of two features from the above process: the notegram and the bass-treble chromagram (refered to as "chromagram" in the following).

Here are some simple showcases of the notegrams, all of which are extracted from acoustic piano audio: Figure 3.3 illustrates the notegram of a middle C

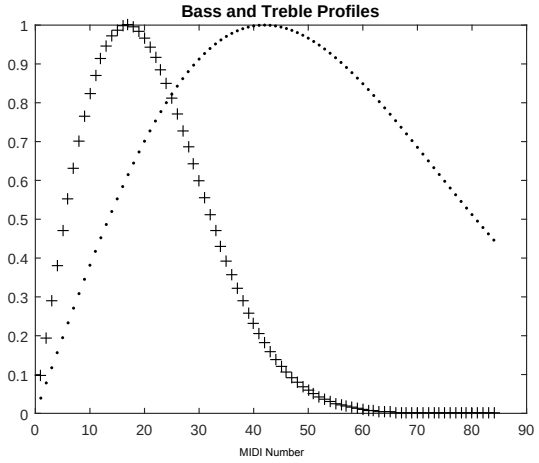

Figure 3.2 Bass (+) and treble (.) profiles. They are both computed in the shape of Rayleigh distributions with scale parameters 16.8 (for bass) and 42 (for treble) respectively. The horizontal axis stands for MIDI pitch numbers. The vertical axis represents normalized profile amplitudes from 0 to 1.

Process	Output level	Bins
STFT	spectrogram	2049
Linear-Log Mapping	log-spectrogram	252
Tuning	notegram (-ns)	252
NNLS	NNLS notegram	84
Bass-treble Profiling	(bass-treble) chromagram (-ch)	24

Table 3.1 Different levels of feature representations

note, Figure 3.4 a major third interval built upon the middle C, and Figure 3.5 a C major chord rooted on the same note. Notice the appearance of harmonics and background noise.

The above are all simple music elements with pure pitches. Figure 3.6 is a notegram example of the intro (first few seconds) of *Let it be*. As this is just a piano intro, no vocal melody is involved. Notice how it is rhythmically segmented.

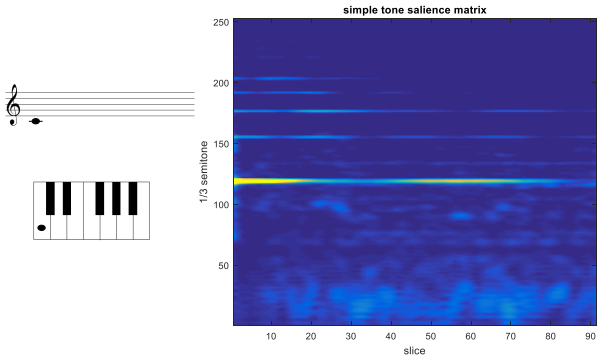

Figure 3.3 Middle C note - notegram

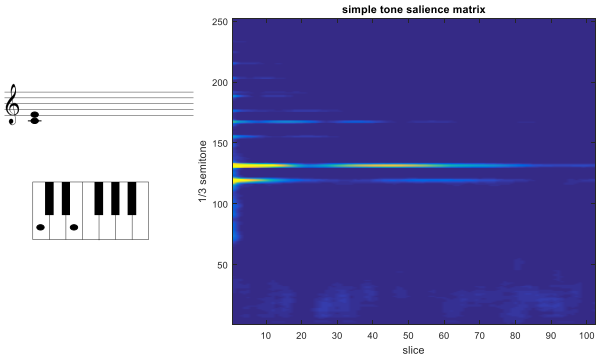

Figure 3.4 Major third interval - notegram

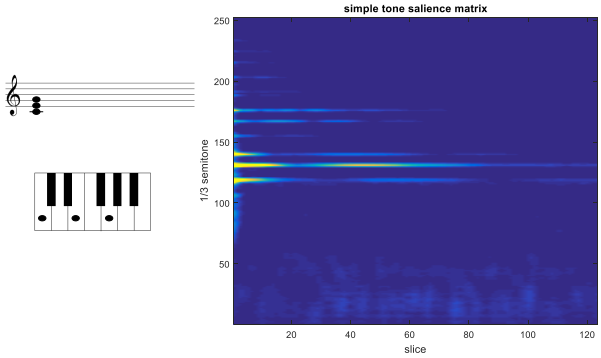

Figure 3.5 C major chord - notegram

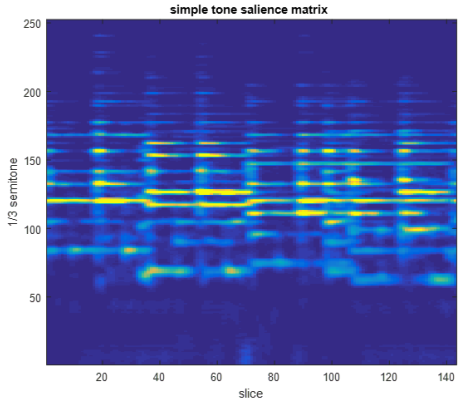

Figure 3.6 Chord progression example - notegram

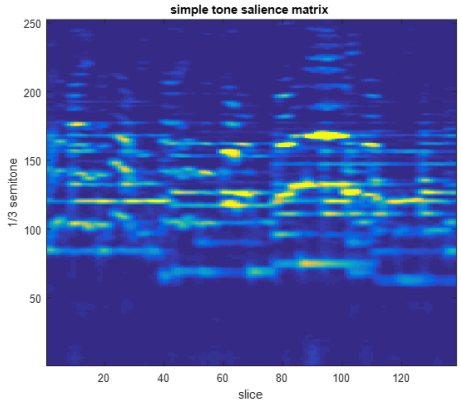

Figure 3.7 Chord progression with vocal melody example - notegram

Figure 3.7 showcases another example with vocal melody. It is extracted from the first line of the first verse of *Let it be*. Notice that although this is much more corrupted, a rhythmical segmentation is also visually noticeable.

Figure 3.8 summarizes the full information flow of the above feature extraction process using the first line of *Let it be* as input. Note how representations are enhanced or compressed during this process.

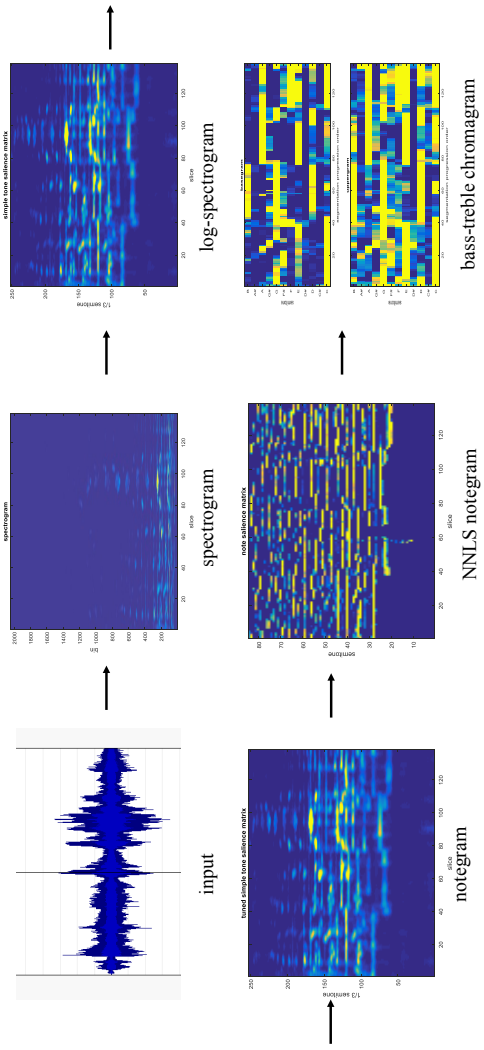

Figure 3.8 Information flow of feature extraction process

3.1.2 Segmentation

The segmentation process is implemented using a GMM-HMM, which is characterized as follows:

- The hidden node models the categorical states of chords. In the *SeventhsBass* vocabulary, there are totally 217 states (1 state per chord), where the 217 is found by multiplying the number of chord types (18)

	μ	σ^2
Bass - chord bass	1	0.1
Bass - not chord bass but chord note	1	0.5
Bass - not bass	0	0.1
Treble - chord note	1	0.2
Treble - not chord note	0	0.2
N.C. (for all notes)	1	0.2

Table 3.2 GMM-HMM segmentation settings. Different parameters are assigned to different note degree within a chord.

with the number of chord roots (12) and adding the number of "N.C." chord types (1).

- The observable node represents a chroma. It is a 24-dimension Gaussian node connecting to the bass-treble chromagram.

- The emission of each hidden state is a 24-dimension Gaussian distribution with parameters specified in Table 3.2. These parameters assign different Gaussians to different pitch classes according to their roles in a chord.

- The transition matrix has heavy uniform self-transition weights, which are 99.99 times of the uniform non-self-transition weight.

- The prior probabilities are uniformly distributed.

3.1.3 Segment Tiling Process

The segment tiling process is introduced to equalize the length of every segment, so as to enable neural networks with fixed-length input. This process divides a segment into N equal-sized sub-segments, and takes a frame-average

within each sub-segment, resulting in an N-frame segment (referred to as Nseg in the following, where N is a variable). If the original number of frames is not divisible by N, the last frame is extended to make it divisible, i.e. this process turns a segment with a variable number of frames into a segment with a fixed number of N frames.

3.1.4 Deep Learning Models

Each Nseg will be classified as a chord label through a deep neural net. There are three types of deep neural nets used here: FCNN, DBN, and RNN.

3.1.4.1 Fully-connected Neural Network

The FCNN is a vanilla neural network with the most basic settings. It is a feedforward neural network, and each layer is fully-connected to the next layer. It applies rectified linear units (ReLUs) as hidden layer activations. It is trained via stochastic gradient descent with back-propagation.

3.1.4.2 Deep Belief Network

The DBN is implemented based on the FCNN. Its multiple hidden layers have sigmoid activations instead of ReLUs.

In the pre-training phase, every pair of adjacent layers (except for the output layer) are trained one pair at a time as restricted Boltzmann machines (RBMs) [HOT06]. In our implementation, the RBM formed by the input layer and the first hidden layer is a Gaussian-Bernoulli RBM, because the input Nseg feature contains real numbers. The RBMs formed by the hidden layer pairs are Bernoulli-Bernoulli RBMs, because each neuron is stochastic binary [HS06].

In the fine-tuning phase, the network is regarded as a feedforward neural network and trained via stochastic gradient descent with back-propagation.

3.1.4.3 Recurrent Neural Network

The RNN shown in Figure 3.9 is bidirectional with long-short-term-memory (LSTM) hidden units [HS97], or a BLSTM-RNN. LSTM is incorporated in order to relieve gradient vanishing/exploding problem for long sequence training [Ben09]. In our LSTM implementation, all gates employ sigmoid activations, while both the cell and output neuron use hyperbolic tangent activations. For a fixed-length Nseg input, the RNN is unrolled into N slices, each handling one input frame. A mean pooling operation is added before the output layer to summarize the LSTM outputs.

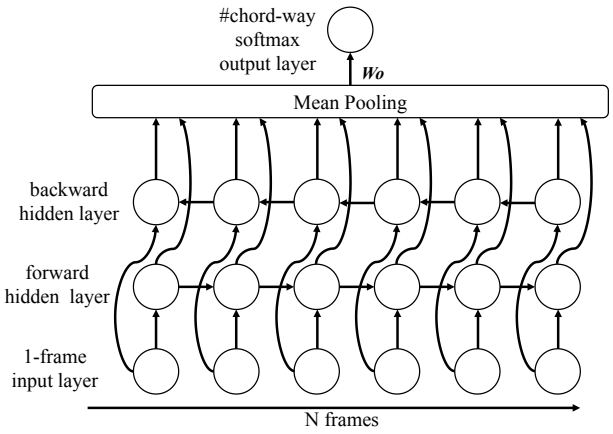

Figure 3.9 The bidirectional recurrent neural network architecture used in the proposed system. Both hidden layers employ LSTM units in place of normal logistic units. The RNN is expanded to N frames, with mean pooling to summarize results.

3.2 Experiments

In this section, we describe a systematic approach to explore and evaluate different system variants of the LVACE framework. We first introduce the datasets, then elaborate the experimental setup, and finally discuss the training and cross-validation (CV) process.

3.2.1 Datasets

For training/cross-validation, we use six datasets of 546 tracks in total. They contain both eastern and western pop/rock songs. They are:

- 29 tracks from the JayChou dataset (JayChou29, or J) [1];
- 20 tracks from a Chinese pop song dataset (CNPop20, or C) [1];
- 26 tracks from the Carole King + Queen dataset (K) dataset [2];
- 191 songs from the USPop dataset (U) [3];
- 100 tracks from the RWC dataset (R) [4];
- 180 tracks from the TheBeatles180 (B) dataset [2].

The datasets are notated by their letter codes. For example, the combination of all datasets is denoted as "CJKURB".

Both chromagram (-ch) and notegram (-ns) are extracted from each track. Both of them can be transposed to all 12 different keys by circular pitch shifting

[1] `http://www.tangkk.net/label/`
[2] `http://isophonics.net/datasets`
[3] `https://github.com/tmc323/Chord-Annotations`
[4] `https://staff.aist.go.jp/m.goto/RWC-MDB/`

(for -ch) or pitch shifting with zero paddings (for -ns). For example, a piece of treble chromagram in key of C can be represented as:

$$PCP_C = (C', C\#', D', D\#', E', F', F\#', G', G\#', A', A\#', B'), \tag{3.5}$$

where X' stands for the salience of pitch class X. It can be circularly shifted to represent an equivalent PCP in other keys, such as key of D:

$$PCP_D = (D', D\#', E', F', F\#', G', G\#', A', A\#', B', C', C\#'). \tag{3.6}$$

As for notegram, although we have pitch saliences instead of pitch class saliences, the same "pitch shifting" ideas can still be applied, given that the out-shifted saliences are filled by zeros.

In practice, the original key is considered as a pivot, and the features are circularly shifted to all 12 keys (the amount of transpositions ranging from -6 to 5 semitones). Adjusting the ground truth chord labels accordingly, this results in a 12-time data augmentation, which helps in reducing over-fitting [Cho14, Hum15].

3.2.2 Experimental Setup

Under the proposed LVACE framework, possible design choices are:

- type of deep neural nets
- depth and width of hidden layers (network configurations)
- number of frames in segment tiling
- input feature representations
- amount of training data

Dimension	Variation
neural net	FCNN; DBN; BLSTM-RNN
segment tiling	1; 2; 3; 6; 9; 12 (seg)
layer depth	2; 3; 4
layer width	500; 800; 1000
input feature	notegram (-ns); chromagram (-ch)
amount of training data	JK; JKU; JKUR; JKURB

Table 3.3 Variations considered in this study

Our study is based on the settings depicted in Table 3.3. For naming conventions: a combination of layer width and depth is denoted as [*width***depth*], such as [800*2]; a segmentation tiling scheme is denoted as *N*seg, such as 6seg; a point in this six dimensional hyper-parameter space is denoted by concatenating each parameter with "-", such as FCNN-6seg-[800*2]-ch-JKU. The space can be explored by parameter sweeping along a given dimension. Particularly, we will first explore along the *layer width* and *layer depth*. We then explore the *segment tiling* scheme with fixed *layer width* and *layer depth*. Following the same strategy, we explore all factors in Table 3.3. This process does not search the whole hyper-parameter space. However, it could gain us some insights of the proposed LVACE framework and produce some good system variants as well.

In this context we regard a "model" as a crossing point of all dimensions in Table 3.3, including training data size. We regard a "system" as a full implementation of the LVACE framework, including the feature extraction, segmentation and deep neural nets. However, since all models share the same feature extraction and segmentation processes, we sometimes use the terms "model" and "system" interchangeably.

3.2.3 Training and Cross-validation

The following training procedures are applied throughout the experiments:

- Each FCNN is trained using mini-batch stochastic gradient descent, and regularized with dropout [SHK+14] and early-stopping [Pre98a].
- Each DBN is pre-trained using contrastive-divergence [HOT06] (CD-10), for 30 epochs with a learning rate of 0.001. It is fine-tuned using the FCNN's training procedure.
- Each BLSTM-RNN is trained using an Adadelta optimizer [Zei12], regularized with dropout and early-stopping.
- All mini-batch stochastic gradient descents use a learning rate of 0.01 and a batch size of 100.
- All early-stopping criteria are monitored using the validation error of the CNPop20 dataset, which is not in any training set. The model with the lowest validation loss will be saved, and if the current validation loss is 0.996 times smaller than the lowest one, the early-stopping patience will increase by the value of the current number of iterations. Training stops when the patience is less than the current number of iterations.
- All dropout rates are set to 0.5.

Five-fold cross-validation (CV) is performed throughout all experiments. Each fold is a combination of approximately 1/5 tracks of each dataset. Every model is trained on four folds and cross validated on the remaining fold, resulting in a total number of five training/cross-validation scores, the average of which will be the final score to be reported.

3.3 Results and Discussions

Throughout this section, we use the MIREX ACE standard evaluation metrics, already introduced in Section 2.3, to report system performances. We report all *WCSR* scores under the *SeventhsBass* evaluation, where a correct classification does *not* involve any chord mapping scheme beyond the *SeventhsBass* vocabulary [PP13]. We use the MusOOEvaluator tool [5] to generate these scores from all the ground truth and predicted chord sequences. Note that we do not report the segmentation score for every system, because they all share the same GMM-HMM process described in Section 3.1.2. This process has a segmentation score of 83% on the JKURB dataset.

In the following discussion, we will analyze the experiment results from a bias-variance perspective [GBD92]. Assuming that a model is trained for multiple times over different samples of the same population, with other settings remain unchanged, the model's prediction for a given input will become a random variable. The model's prediction error can thus be expressed as [FHT01]:

$$Prediction\ Error = Bias^2 + Variance + Irreducible\ Error \tag{3.7}$$

Concretely, a model's bias is defined as the expected difference between its prediction and the ground truth. It measures how much a model's predictions are consistently deviating from the true value, and it indicates whether a model contains fundamentally incorrect assumptions. A model's variance, on the other hand, measures the variance (i.e. the statistical variance, which equals the square of the standard deviation) of the model's prediction. It indicates how much a model's predictions will vary across its different realizations

[5] `https://github.com/jpauwels/MusOOEvaluator`

with different training samples, or equivalently, how much inconsistencies are there within the predictions. Finally, the irreducible error term could be seen as collection of everything that is not bias or variance, such as the noise or inconsistencies in the data itself.

Bias and variance is highly correlated with over-fitting and under-fitting. A high bias model tends to under-fit the data, while a high variance model tends to over-fit the data. A model that neither over-fits nor under-fits, will have low bias and low variance. The amount of bias-variance can be approximated from a model's training and validation (or cross-validation) score:

- A model with high bias (under-fitting) has similar training and validation scores, but none of them is high.
- A model with high variance (over-fitting) has a high training score and a low validation score.

In other words, a model's bias can be approximated as the value of its training or validation error if the two errors are close to each other, and a model's variance can be approximated as the difference between its training and validation errors. Note that the irreducible error could either appear as bias or variance.

In the following, we will examine how each design choice in Table 3.3 actually affects the systems' biases and variances. Moreover, we also compare among different types of deep neural nets and a baseline model, and see which one performs the best in the LVACE task.

3.3.1 Network configurations

Figure 3.10 shows the *WCSR*s of a set of JKU-6seg models with different neural nets, network configurations and input features.

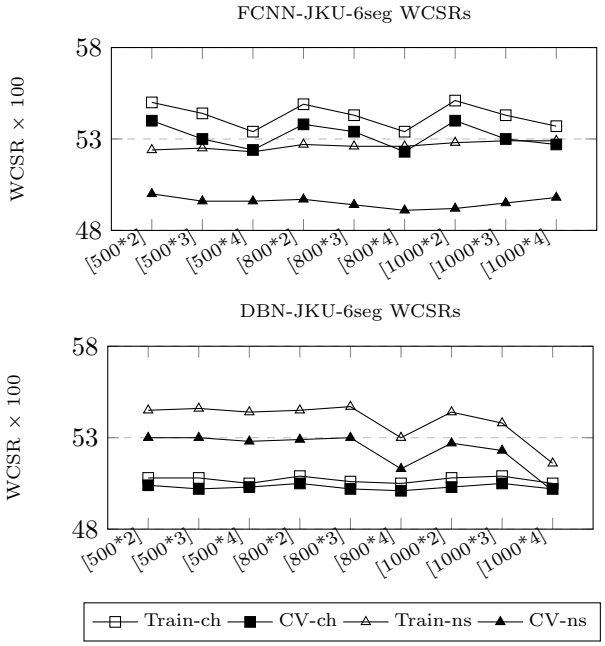

Figure 3.10 Exploring the effect of different network configurations. All models are trained with JKU-6seg-ch.

-ch models — The FCNN has local maximal validation scores when the network has two layers, and it performs worse as network becomes deeper. The DBN's validation scores are stabilized around 50. In this group of experiments, the training and validation scores are close to each other.

-ns models — The FCNN's validation scores are all focused around 50. As for the DBN, there is a trend of performance downgrade along the depth dimension. In both cases the differences between training and validation scores are very small.

Remarks — Firstly, all the FCNN-ch models outperform the FCNN-ns models. This could be largely due to the prior imposed by the chromagram feature. It embeds the knowledge about "pitch classes", and it is originally designed for chord recognition tasks. On the other hand, because the notegram is several

feature transformations away from the chromagram, it contains no such prior information. This could explain why the FCNN-ch models perform worse as they become deeper. Every extra layer will tend to weaken the prior at the input, and at the same time, these layers try to learn some other regularities that map the chromagrams to the chord labels. The results show that, unfortunately, the deeper networks are unable to learn more useful regularities than the prior knowledge already contained in the chromagram.

Secondly, all the DBN-ns models outperform the DBN-ch models (except for the one at [1000*4]). Note that the only difference between the DBN and the FCNN is the generative pre-training process, which in effect is a strong regularization process that prevents over-fitting. This is sometimes equivalent to increasing the model's bias or decreasing the model's variance. As shown in Figure 3.10, since the variances of the FCNN-ch models are already small, the DBN-ch models perform worse than the FCNN-ch models due to the higher biases. However, since the FCNN-ns models have high variances, the DBN-ns models perform better than the FCNN-ns because of the lower variances.

Thirdly, the performance downgrade of the DBN-ns models starting from [800*3] and [1000*2] could be caused by the well-known gradient vanishing problem in deep networks (with sigmoid activations). This could be verified by monitoring the weight updating process. When the gradient vanishing happens, the average amount of weight updates closer to the input will be much less than those closer to the output, resulting in more errors in the earlier layers, which will be aggregated through the feedforward path to the output.

3.3.2 Segment tiling

Figure 3.11 shows the *WCSR*s of a set of JKU-[800*2] models with different neural nets, segment tiling schemes (Nseg) and input features. Note that [800*2] in BLSTM-RNN means there are a forward and a backward hidden layers, each having 800 LSTM units.

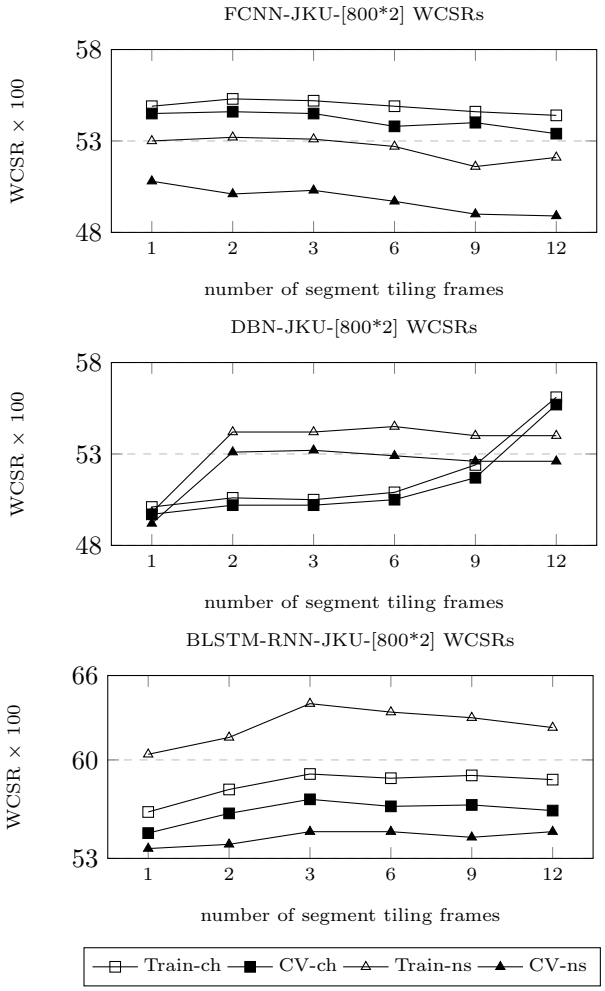

Figure 3.11 Exploring the effect of segment tiling. All models are trained with JKU-ch-[800*2]

-ch models — The FCNN tends to perform worse with larger N; The DBN tends to perform better with larger N; The BLSTM-RNN grows gently when

N is less than 3, and remains relatively constant thereafter. They all have very small variances.

-ns models — Still the FCNN tends to perform worse with larger N; Both the DBN and the BLSTM-RNN have the worst performances when N is 1, and they have higher and stable performances when N is greater than 1.

Remarks — With a large Nseg, a model becomes more complex because of a less blurry input, so that one could expect either less bias, or more variance. In the FCNN models we could observe a tendency of slight variance increasing. This tendency has possibly offset the trend of bias decreasing, which we could not see from Figure 3.11.

For the DBN, as discussed in Section 3.3.1, the generative pre-training processes could reduce the variances or increase the biases. This is clearly reflected in Figure 3.11 if we compare the DBN's *WCSR*s with the FCNN's. On one hand, this could explain why the DBN-ch models have an increasing trend of performances (as well as a decreasing trend of biases) with a larger N, because the DBN-ch model has a much higher bias than the FCNN-ch one when N equals 1. On the other hand, it could also explain why there is a performance boost from $N = 1$ to $N = 2$ in the DBN-ns models, and that the DBN-ns models have consistently lower variances and higher performances than the FCNN-ns models.

For the BLSTM-RNN models, the training and CV curves are much more spread out than those of the FCNN and the DBN. On one hand, the RNN imposes a weight sharing mechanism across the segment tiling frames. This

has an effect of regularization by limiting the number of parameters connecting the input layer to the hidden layer, thus limiting the network's ability to recognize arbitrary dependencies across frames. On the other hand, the RNN also introduces a set of recurrent weights that connect each frame to its next frame. This makes the network more flexible in capturing sequential dependencies between frames. This explains why the training and CV curves are so separated. Particularly, the average training scores of the RNN models are much higher than those of the DBN's and the FCNN's, because the RNN is essentially biased towards problems with sequential natures, and the ACE is one of these problems. Still, we have relatively low CV scores in these RNN models, which lead to high variances. As we will see in the following subsection, this can be remedied by more training data.

3.3.3 Amount of training data

Figure 3.12 shows the *WCSR*s of a set of 6seg-[800*2] models with different neural nets, training data sizes and input features.

For both -ch and -ns models — In all three plots, there are clear trends that increasing the amount of data boosts the models' performances. While the variances of the FCNN and DBN models remain being small, the variances of the BLSTM-RNN models tend to decrease with the increase of data. Interestingly for the FCNN, the increase of data from JKUR to JKURB leads to larger variances and worse CV scores.

Remarks — Similar to Figure 3.11, the FCNN's and DBN's training and CV curves as shown in Figure 3.12 are still very close to each other. It seems that as the amount of data increases, their models have saturated at some

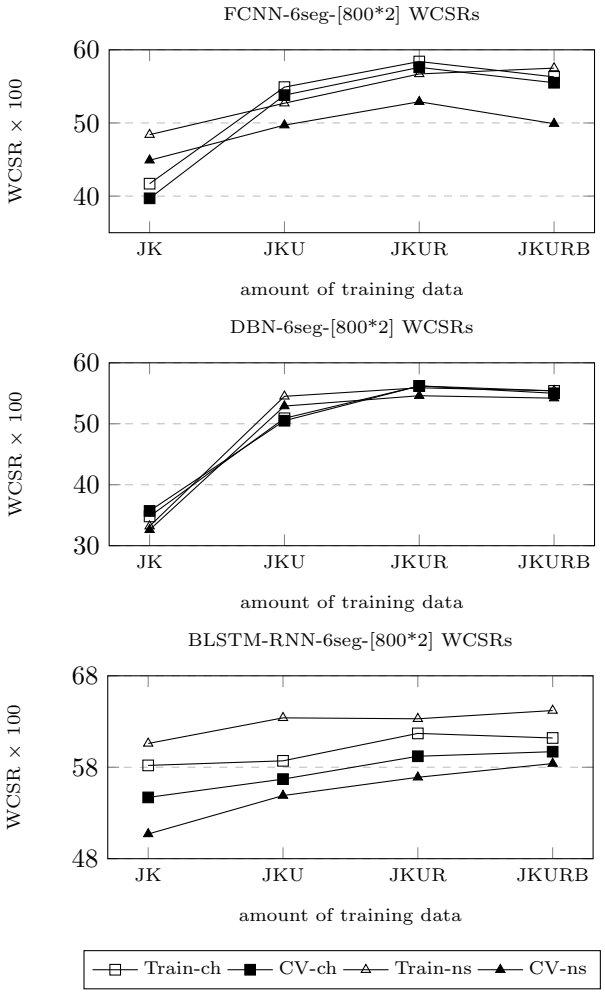

Figure 3.12 Exploring the different training data size. All models are trained with 6seg-ch-[800*2].

point and there is little room for further improvement. On the contrary, the BLSTM-RNN's training and CV curves are much wider apart, and the models tend to generalize better as the data size grows. The results in Figure 3.12 seem to suggest that we have almost touched a performance "ceiling" of the BLSTM-RNN-ch models, but we have yet to reach that of the BLSTM-RNN-ns models.

3.3.4 Input feature

From Figure 3.10 to 3.12, we see that the training and CV curves of the chromagram models are on average much closer than those of the notegram models. This suggests that the prior knowledge contained in the chromagram feature actually introduces bias in the model. On one hand, this may lead to better models if the amount training data is limited (this discussion is not valid for the DBN because of the generative pre-training process). On the other hand, this can also limit the models' improvement when we have sufficient amount of training data. For example in Figure 3.12, we see a potential trend that if more data is added to the model, the BLSTM-RNN-ns model will eventually outperform the BLSTM-RNN-ch model, because the BLSTM-RNN-ns has a higher ceiling (the training score) than the BLSTM-RNN-ch.

3.3.5 Balanced performance

The above discussions are focused on the overall *WCSR*s of different models. Here we are going to examine the models' performances on specific chords. Note that in our datasets (which we believe are good representatives of pop and rock music in general), the chord distributions are highly skewed (as shown in Table 3.3.5), where the maj and min triads make up almost 70% of the whole sample population, the $maj7$, $min7$ and 7 chords constitute more than 20%, and the portion of other chords are less than 10%. In the following discussion, we refer to "common chords" as the maj and min chords, "uncommon chords" as the sevenths chords, including the $maj7$, $min7$ and 7 chords, and "long-tail chords" as all the other chords in the *SeventhsBass* vocabulary. Moreover,

we use “chord” and “chord type” interchangeably to refer to a certain type of chords.

Figure 3.13 shows how different deep neural net models perform on different chords. It is surprising to see that the FCNN and the DBN outperform the BLSTM-RNN only in the *maj* chord category, while the BLSTM-RNN outscores the other two by large margins in most long-tail chords and uncommon chords categories.

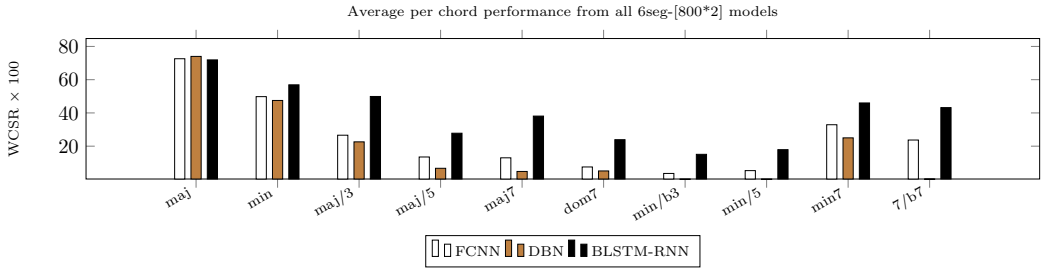

Figure 3.13 Performance on different chords in different neural nets. All models are trained with 6seg-[800*2].

Furthermore, we examine the versatilities of different deep neural net models. We measure them using the *ACQA*, which averages the *WCSR*s of all chords with equal weights. Models that over-fit a few chord types tend to give lower *ACQA*s, while those well-balanced ones will have higher *ACQA*s. As shown in Figure 3.14, the average *ACQA* of the BLSTM-RNN models outscores the average *ACQA*s of the other two types of models by around 10 points.

Dataset	Tracks	maj/5	maj/3	maj	maj7/5	maj7/3	maj7/7	maj7	7/5	7/3	7/b7	7	min/5	min/b3	min	min7/5	min7/b3	min7/b7	min7
C	20	3.2	4.5	38.4	0.2	0.2	0.0	10.2	0.0	0.3	1.1	9.3	2.7	0.0	20.3	0.5	0.0	0.1	9.1
J	29	4.1	8.1	32.3	1.2	0.2	0.1	6.9	0.4	1.5	2.3	5.0	0.7	1.3	15.1	0.7	0.0	0.3	19.8
K	26	4.5	3.9	52.0	0.0	0.3	0.2	5.1	0.3	0.2	0.5	6.2	0.1	0.3	14.9	0.2	0.0	0.8	10.4
U	191	2.3	3.9	54.7	0.0	0.1	0.1	3.2	0.1	0.3	0.3	8.3	0.4	0.4	15.1	0.0	0.1	0.4	10.2
R	100	2.5	4.6	42.8	0.3	0.1	0.1	8.9	0.0	0.2	0.3	7.9	0.4	0.5	15.3	0.0	0.1	0.2	15.7
B	180	2.4	1.0	66.1	0.0	0.2	0.3	0.9	0.1	0.1	0.4	8.7	0.6	0.5	15.9	0.0	0.1	0.4	2.5
Weighted Average		2.6	3.3	54.3	0.1	0.1	0.2	4.0	0.1	0.3	0.5	8.1	0.6	0.5	15.6	0.1	0.1	0.4	9.1

Table 3.4 Distribution of chords in the datasets. (maj and min: 69.9%; maj7, min7 and 7: 21.3%; others: 8.8%)

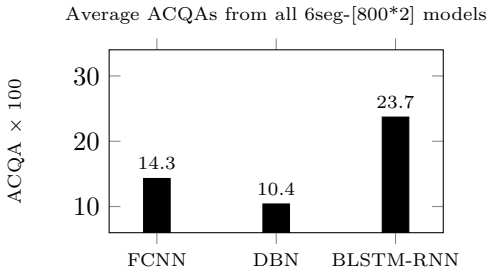

Figure 3.14 Average ACQAs of *SeventhsBass* Vocabulary. All models are trained with 6seg-[800*2].

We perform a Friedman test [Fri37] on the track-wise $ACQA$ results. After that we use the Tukey HSD (honest significant difference) [Tuk49] to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. As shown in Figure 3.15, both BLSTM-JKURB-ch-6seg-800 and BLSTM-JKURB-ns-6seg-800 are significantly better (no overlap of confidence intervals) than the other systems, and BLSTM-JKURB-ch-6seg-800 is significantly better than BLSTM-JKURB-ns-6seg-800 as well. This concludes that the BLSTM-RNN models are significantly better than the FCNN and the DBN models in terms of $ACQA$s, or balanced performances.

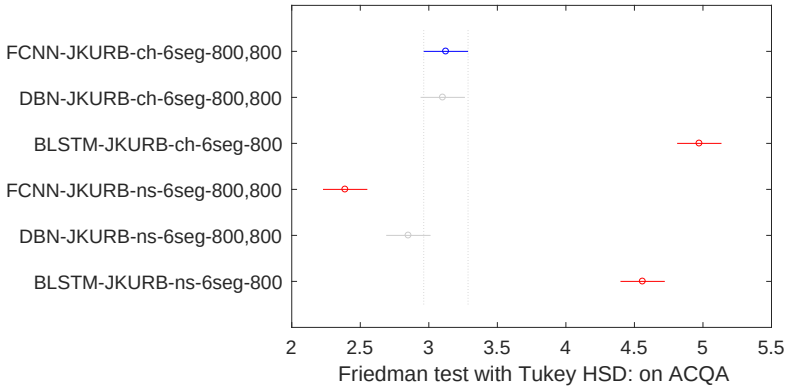

Figure 3.15 Friedman test with Tukey HSD: ACQAs of different system variants

Now we have concrete evidence that the BLSTM-RNN is a better neural network in solving the LVACE problem than the other two models. It is reasonable to think that the BLSTM-RNN regards its input as a *sequence of frames*, while fully-connected networks (in this context the FCNN and DBN) regard their inputs as *flat vectors*. Therefore, while the BLSTM-RNN tries to look for regularities within *each pair of consecutive frames* along the time direction, the FCNN or DBN would search for regularities within *every point of the flat vector* as if they are not time related at all. Another perspective is that the BLSTM-RNN has $1/N$ times the weights as much as those of the FCNN's and the DBN's between the input layer and the first hidden layer. Thus the weight sharing over multiple frames prevents the BLSTM-RNN from over-fitting, and allows the model to process higher resolution inputs without an increase in parameters.

In some cases, the fully-connected network is more efficient given that the input feature has already encoded certain prior information about music (e.g. chromagram contains the information about pitch classes). Nevertheless, it overlooks the "sequential order of frames", which probably causes the over-fitting of root position chords and the under-fitting of chord inversions.

3.3.6 Baseline comparison

Finally, we compare our LVACE framework with the Chordino. It should be emphasized that Chordino is the best suitable baseline because: (1) Our framework resembles Chordino in terms of the segmentation and feature extraction processes; (2) Chordino is the only other system that supports seventh chords and chord inversions; (3) Chordino perform segmentation and classification in one single pass. By comparing with the Chordino, we could have a

good idea how separating the segmentation and classification affects system performance.

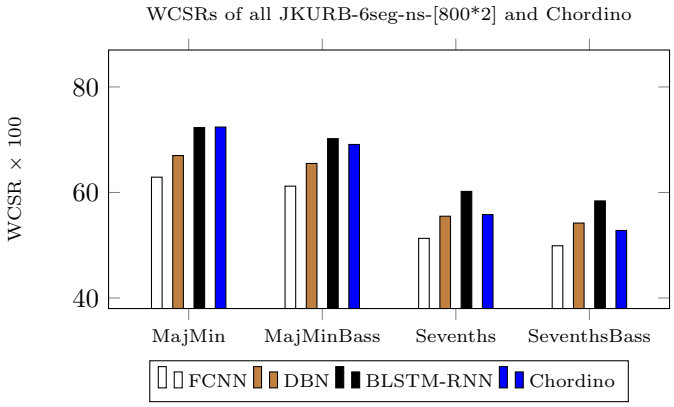

Figure 3.16 Performance comparison between system representatives and Chordino on WCSRs. All models are trained with JKURB-6seg-ns-[800*2].

We choose one representative for each type of deep neural net, all trained and cross-validated with JKURB-6seg-ns-[800*2], and compare them with the Chordino using the standard MIREX ACE categories. As shown in Figure 3.16, the representative of BLSTM-RNN outperforms the Chordino by large margin in *Sevenths* and *SeventhsBass*, and it scores fairly close to the Chordino in *MajMin* and *MajMinBass*. The other two representatives are not performing as good as the Chordino in most categories.

We perform a Friedman test on the track-wise *SeventhsBass WCSR* results. After that we use the Tukey HSD to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. As shown in Figure 3.17, BLSTM-JKURB-ns-6seg-[800*2] is significantly better than the other systems as well as the Chordino.

In terms of *ACQA*, as shown in Figure 3.18, Chordino outperforms both the FCNN's and DBN's representatives, but the most balanced system is the

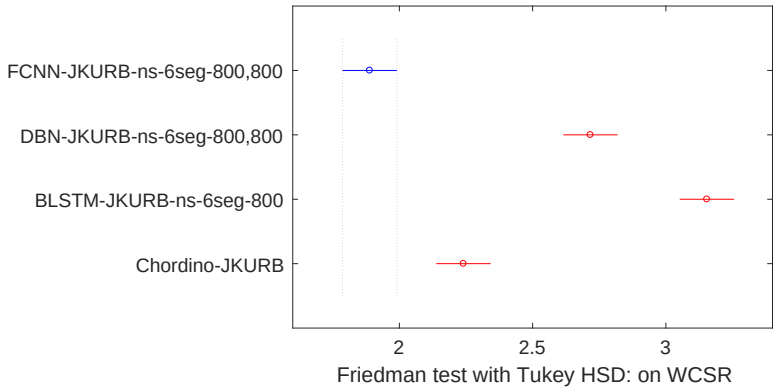

Figure 3.17 Friedman test with Tukey HSD: WCSRs compared with the baseline

BLSTM-RNN's representative. We again perform a Friedman test with Tukey HSD ($p = 0.05$, using the track-wise results) to test whether the differences in *ACQA*s are significant. As shown in Figure 3.19, the BLSTM-JKURB-ns-6seg-[800*2] system is again significantly better than the other systems as well as the Chordino.

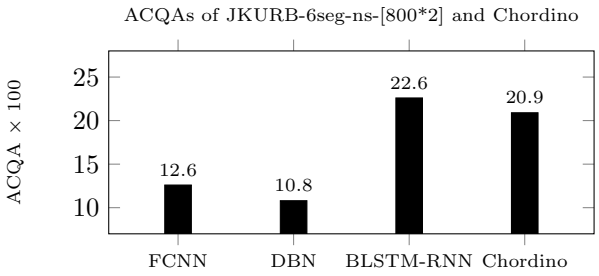

Figure 3.18 Performance comparison between system representatives and Chordino on ACQAs. All models are trained with JKURB-6seg-ns-[800*2].

3.4 MIREX 2016 Results

MIREX ACE 2016 features 4 sets of systems implemented by four different groups:

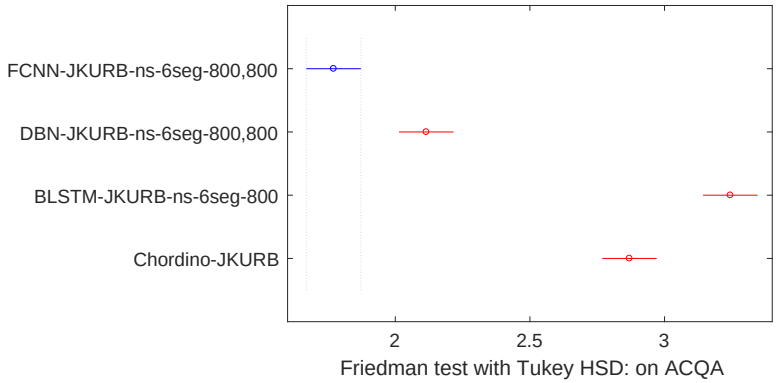

Figure 3.19 Friedman test with Tukey HSD: ACQAs compared with the baseline

- CM1: Chordino, from Queen Mary University of London;
- DK*: features three variants of the LVACE approach proposed in this chapter [DK16] (note that DK4, which is from another design approach, is skipped in the following discussion);
- FK*: implemented by Korzeniowski and Widmer, featuring the systems in two of their works [KW16a, KW16b]
- KO1: shineChords, a time-frequency reassign approach proposed by Khadkevich and Omologo [KO11].

3.4.1 Dataset and Vocabulary

It should be noted that except for CM3, whose model parameters are all manually specified, all other systems involve extensive training. Due to data scarcity in ACE, some of the test data in MIREX ACE are also used for training by some systems.

KO1's description does not mention its training set, but it is probably trained with the Isophonic dataset (MIREX 2009 set) [BdHP14]. This can

be inferred by noticing that, compared with other systems, it has too high a *SeventhsBass* score in Isophonic, while not as high in other test sets.

FK* are trained with the Isophonic, Robbie Williams [6], RWC and the public part of McGill Billboard [7] dataset.

DK* are trained with the USPop and RWC dataset, which are not part of MIREX ACE 2016's test sets. Note that DK1 is the DBN-6seg-ch-[800,800]-UR system and DK2 is the BLSTM-RNN-6seg-ch-[800,800]-UR system.

In terms of chord vocabulary, both DK3, FK* and KO1 only support *majmin*; CM1 supports a large vocabulary including most types in *SeventhsBass*; and DK1&2 support exactly the *SeventhsBass*.

3.4.2 Results and Discussions

Table 3.5 shows the MIREX ACE 2016 results (for brevity, in the tables, B = *Bass*, Mm = *MajMin*, MmB = *MajMinBass*, R = *Root*, S = *Sevenths*, SB = *SeventhsBass*, Seg = *Segmentation Quality*). Within the large vocabulary context of this thesis, the main focus is the *SeventhsBass* column.

In the Isophonic test, KO1 gets the highest score. This is probably because of its over-fitting of the data, as explained previously. Note that it scores much lower in tests that it has not previously seen, such as JayChou 2015 (The same as the JayChou29). Both FK2 and FK4 come next to KO1, but unfortunately they are all informed with this test set during training. Since it is not clear to what extend the systems over-learn the data, the results of this test is not statistically valid to deduce any reasonable conclusions.

[6] `https://www.researchgate.net/publication/260399240_Chord_and_Harmony_annotations_of_the_first_five_albums_by_Robbie_Williams`

[7] `http://ddmal.music.mcgill.ca/billboard`

Algorithm	R	Mm	MmB	S	SB	Seg	UnderSeg	OverSeg
Isophonics 2009								
CM1	78.56	75.41	72.48	54.67	52.26	85.90	87.17	86.09
DK1	79.21	76.19	74.00	66.02	64.15	85.71	82.62	91.23
DK2	77.84	74.49	71.93	61.61	59.47	85.82	82.72	91.28
DK3	80.03	77.55	74.79	68.40	65.88	85.81	82.50	91.53
DK4	76.05	72.96	71.41	62.77	61.44	78.19	87.97	72.43
FK2	86.09	85.53	82.24	74.42	71.54	87.76	85.79	90.73
FK4	82.28	80.93	78.03	70.91	68.26	85.62	82.40	90.89
KO1	82.93	82.19	79.61	76.04	73.43	87.69	85.66	91.24
Billboard 2012								
CM1	74.15	72.22	70.21	55.35	53.40	83.64	85.31	83.39
DK1	75.28	73.57	71.87	59.98	58.53	83.35	80.26	88.52
DK2	73.77	71.69	69.86	58.66	57.00	83.57	80.40	88.70
DK3	75.92	74.75	72.69	53.42	51.67	83.39	79.97	88.92
DK4	72.59	70.85	69.78	56.29	55.36	76.13	87.72	70.05
FK2	85.64	85.38	82.55	60.70	58.38	87.62	86.09	90.13
FK4	79.23	78.62	76.20	56.53	54.51	85.09	81.98	89.94
KO1	77.45	75.58	73.51	57.68	55.82	84.16	82.80	87.44
Billboard 2013								
CM1	71.16	67.28	65.20	48.99	47.17	81.54	83.11	82.63
DK1	72.06	68.69	67.26	54.54	53.29	80.82	77.58	88.06
DK2	70.18	66.54	64.66	52.97	51.41	80.85	77.68	88.02
DK3	72.39	68.53	66.55	48.99	47.28	80.76	77.26	88.30
DK4	69.56	65.83	64.78	51.81	50.93	74.55	86.31	69.18
FK2	80.07	77.89	75.42	55.41	53.22	82.94	82.43	86.80
FK4	74.66	71.85	69.44	51.93	49.80	80.61	77.19	88.70
KO1	75.36	71.39	69.43	53.57	51.78	81.63	79.61	87.75
JayChou 2015								
CM1	72.75	72.08	65.48	54.39	48.98	86.60	86.89	86.91
DK1	74.70	73.87	70.33	54.98	52.25	86.76	82.78	91.79
DK2	72.19	72.55	69.10	54.09	51.46	87.09	83.35	91.75
DK3	75.01	74.75	63.56	49.27	40.24	86.76	82.54	92.08
DK4	71.51	69.03	65.93	50.07	47.45	78.11	87.87	70.56
FK2	79.51	78.66	68.15	50.69	42.34	86.81	85.43	88.56
FK4	76.13	75.44	64.36	49.69	40.74	84.55	81.22	88.95
KO1	78.73	77.69	66.87	54.16	44.55	88.46	87.12	90.11
RobbieWilliams 2016								
CM1	81.90	78.25	76.05	57.92	55.90	87.96	88.96	87.45
DK1	81.50	77.77	76.10	68.88	67.34	87.03	83.22	92.11
DK2	79.01	75.97	73.57	65.26	62.98	87.20	83.40	92.23
DK3	81.85	78.56	76.16	74.71	72.55	86.98	82.95	92.34
DK4	78.92	75.15	73.66	66.72	65.34	81.82	88.44	76.88
FK2	88.53	87.23	84.19	82.57	79.88	90.04	88.62	91.88
FK4	83.37	80.96	78.42	77.04	74.76	87.22	84.50	91.02
KO1	83.55	80.33	78.16	73.54	71.39	88.04	85.39	91.68

Table 3.5 MIREX 2016 Results

In Billboard 2012 and 2013 tests, the best performances are both from DK1. Although FK* learn from part of the test set, they somehow does not outperform DK1. Moreover, DK1 outscores KO1 by about 3 points in both tests.

JayChou 2015 test is probably the most convincing one, since none of the systems are learning this set. Moreover, more than 20% of it are chords other than *maj*, *min* and *sevenths*. This portion is much more than those in the other test sets. The performance on this test could well demonstrate a system's vocabulary versatility. This can be noticed by making comparisons among the DK* systems, where DK1 and DK2 perform much better than DK3 in *SeventhsBass*. The best two systems in this test are DK1 and DK2, outscoring the FK* systems by around 10 points, and the KO1 system by around 8 points.

Robbie Williams test somehow shows the reverse ranking of JayChou 2015 test. DK3 leads both DK1 and DK2 by about 5 and 10 points. FK* are the best performing ones, but they might as well over-fit the data. It is interesting to note that KO1 scores 71.39, slightly less than DK3's 72.55. This strongly indicates that this test set is mainly composed of *maj*, *min* and *sevenths* (probably $> 90\%$), which is very similar to the chord composition of the Isophonic set. Therefore systems that only support a small chord vocabulary will have a big advantage by not being able to confuse *maj* and *min* chords with the long-tail chords.

3.5 On Smaller Vocabularies

This chapter is mainly developed for large vocabulary, specifically, the *SeventhsBass* vocabulary. In order to do a thorough comparison between the proposed system framework and other approaches, a set of experiments are also conducted using two smaller vocabularies, i.e., 1, *MajMin* (the same as *majmin*); 2, *Full121* vocabulary (already introduced in Section 2.2.6).

For *MajMin* implementation, each label in the training set will be mapped to its *MajMin* form using a normal chord mapping scheme [Har10, PP13]. As a result, each deep neural net will be trained on a dataset with only *MajMin* labels. For *Full121* implementation, we use Mauch's chord mapping strategy [Mau10]. In the following all systems are implemented to support exactly the *MajMin* or the *Full121*.

3.5.1 On MajMin

Table 3.6 shows the results, where all systems, except for Chordino, only support the *MajMin* vocabulary. On one hand, the overall performances of these systems are only fairly comparable with those in Section 3.3.6, on the other hand, the *ACQA* of these systems are all much lower because of a much smaller vocabulary than Chordino's.

Table 3.7 shows the *SeventhsBass* categorical breakdown. Take DBN-ch as a representative, since it only supports *MajMin*, its *maj* and *min* scores are much higher than the Chordino's. As the two chord types take up the majority of the dataset's population, DBN-ch has a much higher *SeventhsBass* score in Table 3.6. However, by relaxing the evaluation strength from *SeventhsBass*

System	B	Mm	MmB	R	S	SB	Seg	ACQA
Chordino	76.41	74.30	71.40	77.19	52.99	50.60	83.87	16.61
FCNN-ch	74.71	73.25	71.22	75.74	65.08	63.18	83.57	8.42
DBN-ch	77.04	75.50	73.39	78.10	67.26	65.30	83.78	8.71
BLSTM-RNN-ch	76.88	75.05	72.84	78.11	66.58	64.50	83.74	8.75
FCNN-ns	72.65	70.00	68.14	73.35	62.72	60.98	83.59	7.81
DBN-ns	72.91	70.96	69.14	73.66	63.33	61.66	83.44	8.05
BLSTM-RNN-ns	76.04	73.98	71.99	76.85	65.85	64.03	83.76	8.54

Table 3.6 Overall WCSR scores; All systems, besides Chordino, are trained with CJKUR-[800*2], tested on the TheBeatles180 dataset, with only MajMin vocabulary support.

to *Sevenths* and to *MajMin*, the performance difference between DBN-ch and Chordino becomes much smaller.

%B	2.01	0.95	63.31	0.02	0.17	0.27	0.82	0.08	0.06	0.39	8.33	0.61	0.44	14.99	0.01	0.06	0.41	2.37	4.63
	M/5	M/3	M	M7/5	M7/3	M7/7	M7	7/5	7/3	7/b7	7	m/5	m/b3	m	m7/5	m7/b3	m7/b7	m7	N
Chordino	19.9	17.1	54.4	0.0	0.0	0.0	55.6	0.0	0.0	5.7	41.0	0.0	0.0	54.3	0.0	0.0	0.0	51.0	2.2
FCNN-ch	0.0	0.0	77.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	74.0	0.0	0.0	0.0	0.0	2.8
DBN-ch	0.0	0.0	80.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	76.8	0.0	0.0	0.0	0.0	3.0
BLSTM-RNN-ch	0.0	0.0	79.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	78.2	0.0	0.0	0.0	0.0	2.5
FCNN-ns	0.0	0.0	76.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	64.2	0.0	0.0	0.0	0.0	2.9
DBN-ns	0.0	0.0	76.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	68.6	0.0	0.0	0.0	0.0	2.9
BLSTM-RNN-ns	0.0	0.0	79.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	74.7	0.0	0.0	0.0	0.0	2.7

Table 3.7 Detail SeventhsBass WCSR scores. All systems, besides Chordino, are trained with CJKUR-[800*2], with only MajMin vocabulary support. M = major, m = minor, N = N.C (no chord). The %B row shows the composition of chords in the test set.

3.5.2 On Full121

Table 3.8 shows the *WCSR*s of *Full121* vocabulary with the same TheBeatles180 test set. Table 3.9 and 3.10 show the *Full121* scores by other systems using a slightly larger 216-track Isophonic test set, which is extended from the TheBeatles180 test set. Table 3.9 is reported in a 3-fold cross-validation manner, while Table 3.10 is in 5-fold.

System	WCSR	Seg
FCNN	73.43	83.77
DBN	75.46	83.92
BLSTM-RNN	73.63	83.80

Table 3.8 Full121 scores. Tested on the TheBeatles180 dataset. All systems are trained with CJKUR-[800*2]-ch, with Full121 vocabulary.

The computational processes of both *CP* and *RCO* metrics are essentially similar to that of the *WCSR*, and they only accept "exact chord match" under the vocabulary [NMSRDB12, Mau10].

System	Chord Precision (CP)	Seg
CH (Chordino)	50.31	N/A
HP (Harmonic Analyzer)	70.26	N/A

Table 3.9 Full121 scores. Tested on the Isophonic 2009 dataset with 3-fold cross-validation[NMSRDB12]

It is difficult to make any conclusive statement out of these three different tables, which are from slightly different test sets, and reported in different ways. But it is reasonably fair to say that the proposed system framework is not bad at all in handling the *Full121* vocabulary.

System	Relative Correct Overlap (RCO)	Seg
trained MBK	41.8	N/A
MBK	56.7	77.9
MBK-NNLS	61.8	N/A

Table 3.10 Full121 scores. Tested on the Isophonic 2009 dataset with 5-fold cross-validation [Mau10]

3.6 Summary

This chapter presents an in-depth discussion of a hybrid GMM-HMM and deep neural nets LVACE approach. The study is motivated by a current research gap in ACE, i.e. the general negligence of segmentation-classification separation approach and large vocabulary system. This serves as the basis for the proposed chord inversions supportable segmentation-classification separated LVACE system framework. The system framework leverages handcrafted feature extraction and segmentation processes, and plugs in various deep neural nets to classify the chord within each segment. Experimental results indicate that:

- The chromagram feature contains prior knowledge about musical pitch class that increases the bias and limits the potential improvement of the models.

- The BLSTM-RNN can learn regularities from the notegram feature that potentially outperforms the chromagram feature.

- The BLSTM-RNN's representative system (with all available training data) has significantly better *WCSR* and *ACQA* than the FCNN's one, the DBN's one, and the Chordino.

Despite the best system variant significantly outperforms the baseline system, all training and CV scores presented in this chapter are still far less than 100%. This indicates either there is large bias in the LVACE framework itself, or there is irreducible error in the underlying data. We speculate three potential causes as explained in the following.

Firstly, the performance of the proposed framework is upper-bounded by the segmentation performance of the GMM-HMM process introduced in Section 3.1.2, and the performance of this process on the JKURB set is 83%.

Secondly, the segment tiling process introduces bias to the system, since it assumes a chord can be correctly recognized after we tile its original features into several frames of averaged features. This process could help prevent over-fitting by regularizing the degree of freedom of the input, but at times it scarifies important information conveyed in the original variable-length features.

The above two points set a hard performance limit of the proposed LVACE framework: unless the chord segmentation technique is perfect and the segment tiling process is completely excluded, one could not expect a system with very low bias.

Thirdly, there is non-negligible amount of noise in the ground truth annotations themselves. Inevitably, due to differences in musical training, human annotators sometimes disagree, particularly on long-tail chords [HB15]. This results in a glass ceiling for LVACE: unless there are more data for uncommon and long-tail chords and they are more consistently labeled, all efforts for improving LVACE will be hindered by the lack of skewed class training and data consistency.

In a very strict sense, there is not any "gold standard" if human annotators themselves might disagree with each other. But in a loose sense, there could be a "gold standard" if:

- all annotations are done by only one annotator, or
- all annotations are done by multiple annotators (much more than two).

In the former case, the only annotator "dictates" a local "gold standard", so that whenever a machine tries to learn from the data, it actually targets at this annotator's "style". In the latter case, multiple annotators decide a "gold standard" in a way such as majority vote or data fusion [KdHV15, Kle04], so that a trained model actually aims at the optimal "style" that minimizes the objections among these annotators. Therefore, although the "gold standard" is indeed an important issue, we still have to design a system that "learns well".

We believe that the next step of LVACE research should focus more on improving the recognition accuracies on uncommon and long-tail chords. That is, instead of considering the overall *WCSR* of a large vocabulary, attention should also be given to the balanced metric, such as *ACQA*. Although we have pointed out that the BLSTM-RNN is very promising in handling large vocabulary with inversions, we have yet to explored possible ways to train the network under such "imbalanced class population " scenario [CJK04]. More importantly, we should spend greater efforts on data collection, particularly of long-tail chords, and at the same time ensure the data integrity and consistency, in the future development of LVACE.

Finally, as a potential application, a piece of timed chord sequence output can be combined with the *.lrc* file [8] of the same track to become a chord-lyrics sheet, which is one of the most frequently used pop music sheet formats. Figure 3.20 shows an example output of this automatic process [9]. Note that in this case the lyrics might not be perfectly aligned with chords, since an *.lrc* file only has information on the start and end time of each line, instead of each word. A more sophisticated solution would be to use the audio-lyrics alignment tool [MFG10] as a preprocessing stage before the chord-lyrics matching.

[8] `https://en.wikipedia.org/wiki/LRC_(file_format)`
[9] `https://github.com/tangkk/songtranspose`

```
歌名: 愛很簡單
C#:maj |
演唱: 陶喆
作詞: 娃娃
作曲: 陶喆
C#:maj | F:min |
忘了是怎麼開始
A#:min |
也許就是對你
C#:maj/5 | F#:maj | G#:maj |
有一種感覺
C#:maj | F:min |
忽然間發現自己
A#:min |
已深深愛上你
C#:maj/5 | F#:maj | D#:min |
真的很簡單
A#:min | F:min |
愛的地暗天黑都已無所謂
C:7 | D#:min |
是是非非無法抉擇  喔~~
A#:min | F:min |
沒有後悔為愛日夜去跟隨
F#:maj | G#:7 |
那個瘋狂的人是我  喔~~
C#:maj | G#:maj/3 | A#:min |
I Love You
C#:maj/5 | F#:maj | C#:maj/3 |
無法不愛你 Baby
D#:min7 | G#:maj |
說你也愛我  喔~~
C#:maj | G#:maj/3 | A#:min | C#:maj/5 |
I Love You
F#:maj |
永遠不願意 Baby
G#:7 | C#:maj | C#:maj |
失去你
```

Figure 3.20 Chord-lyrics output of *Aihenjiandan* (by Taiwan singer-songwriter David Tao)

Chapter 4

A Recurrent Neural Network Approach with Even Chance Training

As recapped in Section 2.2.5, by far there are four types of ACE approaches:

- local feature extraction - global smoothing [Fuj99, SE03]
- local classification - global smoothing [HB12];
- local feature learning - global classification - global smoothing [BLBV13, SBLD15];
- local feature learning - local classification - global smoothing [ZL15].

All of these approaches incorporate a "global smoothing" stage after classification. This is a step that requires prior domain knowledge which does not belong to the deep learning framework. From a machine learning perspective, prior knowledge is considered suboptimal, because it is imposed by human engineers rather than learned by machines. It could also be suboptimal in terms of performance, if the machine learning model is powerful or the amount of training data is large enough. A machine learning system evolves by reducing the amount of prior knowledge involved while maintaining the overall system performances as much as possible.

Moreover, all existing approaches overlook the fact that the chords are distributed in a highly uneven way and that the chord classification problem is actually a skewed classes classification problem. If the similar approaches are used for LVACE, it is very probable that the systems will over-fit the chord types that are highly over-represented.

This chapter presents an approach that on one hand employs a new neural network training scheme to achieve a more balanced performance on both common and uncommon chords, and on the other hand does not rely on "global smoothing".

Under the above taxonomy, the approach presented in this chapter belongs to a new **"local feature extraction - global classification"** category. Instead of feature learning, it employs a normal feature extraction process, and the segmentation and classification processes are all done by one single RNN pass. To accommodate to large vocabulary, the RNN is trained via a skewed class oriented scheme. This scheme is shown to be very effective in improving the system's vocabulary versatility, compared with a scheme which does not attend to the skewed chord distribution.

4.1 System Overview

Figure 4.1 shows the system overview, which mainly contains a feature extraction module and a BLSTM-RNN sequence decoding module. The former is the same as the feature extraction process presented in Chapter 3.

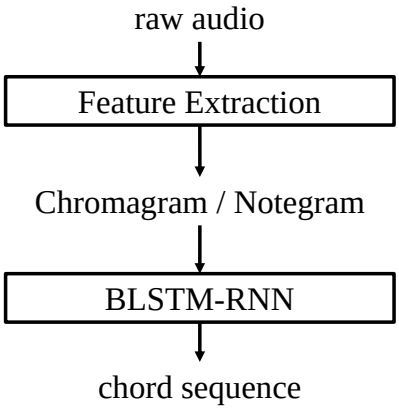

Figure 4.1 BLSTM-RNN LVACE System Overview. The raw audio is feature extracted into a piece of chromagram or notegram, and then decoded by a BLSTM-RNN into a segmented chord sequence.

4.2 Implementation

The key module in this system is the BLSTM-RNN. This section will first briefly discuss RNN's variable-length sequence training, and then elaborate on the implementation of the BLSTM-RNN.

4.2.1 RNN Training

An RNN can model the conditional probability of a label sequence l given its corresponding input feature sequence x. Let's define the label alphabet as L, and let both x and l have the same length T in the unit of time. The conditional probability $p(l|x)$ modeled by the RNN can be written as:

$$p(l|x) = \prod_{t=1}^{T} y_{l_t}^t \qquad l \in L \quad and \quad |l| = |x| \tag{4.1}$$

where l_t is the t^{th} element of l and $y_{l_t}^t$ is the probability of predicting label l_t at time t given x. At time t, $y_{l_t}^t$ can also be written as $p(l_t|x)$, which is the value of the l^{th} slot of RNN's prediction at time t. This RNN can be trained using

a maximum likelihood approach by updating the network weights towards a direction that increases the conditional probability depicted in Equation 4.1. Thus this equation, or its logarithm, can be used as the objective function (cost function).

4.2.2 BLSTM-RNN Architecture

Figure 4.2 shows the graphical model of the BLSTM-RNN used in this system framework, which has a forward and a backward hidden layer both with 800 LSTM units. The input layer has the same dimension as the input feature's. The output layer is a *#-chord-way* softmax layer. It takes a sequence of features as input, and generates a sequence of chord labels.

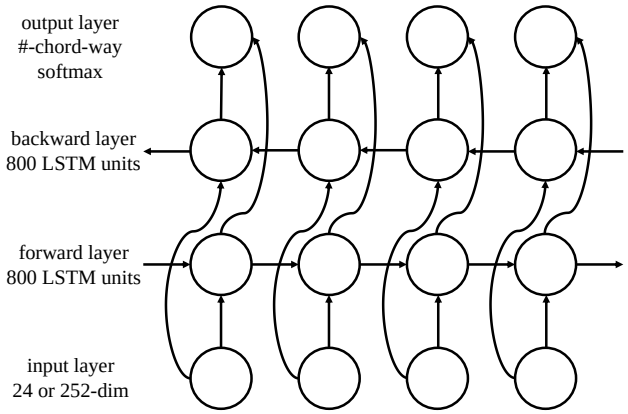

Figure 4.2 The BLSTM-RNN. Both the forward and backward hidden layers contain 800 LSTM units

4.2.3 Vocabulary and Datasets

The large vocabulary supported by the proposed systems is again the *SeventhsBass*. The datasets used in the experiments are the same as the Chapter 3's, but the data generation and augmentation schemes are different, which will be described as follows.

4.2.4 Data Augmentation

To generate training data, firstly all raw audios are transformed to chromagram and notegram representations. The original segment-wise ground truth annotations are upsampled to become frame-wise annotations with 1-to-1 mappings to their input representations. Due to the absence of phase information in chromagram and notegram, all data can be circularly transposed to 12 keys to yield 12 times the original amount of data [HB12].

4.2.5 Training and Cross-validation

Two different training schemes are used. The only difference between them are the way of choosing training cases at each iteration:

- completely random (CR): a random training case is chosen.
- even chance (EC): a training case starting with a certain chord type is chosen, and each chord type has an even chance to be chosen as the start.

The EC training scheme is inspired by the skewed class sensitive training methods [CJK04]. Considering a skewed distribution of chords in the training datasets [BWF11], a random sampling scheme like CR will inevitably draw

samples based on that same distribution, which causes lack of exposure of uncommon chords. The EC scheme, however, gives each class an equal chance of exposure during the training process. Concretely, the EC scheme is formalized as follows in Algorithm 1:

Algorithm 1 EvenChanceTraining

Require: training data set - (X, y); number of chord classes - $nclass$; early stopping flag - es.

od = BalancedOrderedDict(y, $nclass$)
$iter = 0$
while not early-stopping **do**
  if mod($iter$, $nclass$) is 0 **then**
    $coidx$ = random_shuffle(0:$nclass$-1)
  $tclist = od(coidx_{mod(iter,nclass)})$
  draw a random item e from $tclist$
  update network with $(X, y)_e$
  $iter$++

The core of this procedure is the "*BalancedOrderedDict*" which generates a dictionary of *(track index, chord change position)* tuples indexed by chord classes. It is formalized in Algorithm 2, where each entry of *od* contains a list of *(track index, chord change position)* tuples.

The following describes the remaining training procedures that apply throughout the experiments. We try to report the precise settings of every parameter so that the readers may reproduce the results:

- Each training case contains 500 frames of audio content with ground truth labels;

Algorithm 2 BalancedOrderedDict

Require: labels of training data set - y; number of chord classes - $nclass$.

```
for each class i from 0 to nclass − 1 do
    initialize an empty list od[i]
for each track index j in y do
    for each frame poistion k in y[j] do
        if k is a chord change position then
            append (j,k) to od[y[j][k]]
return od
```

- The network update signal is computed by an Adadelta optimizer [Zei12];
- The training is regularized with dropout [SHK+14] and early-stopping [Pre98b];
- All dropout probabilities are set to 0.5;
- All early-stopping criteria are monitored using the validation error of the CNPop20 dataset, which is not in any cross-validation set; The validation cycle is 100 iterations;
- The model with the lowest validation loss will be saved; If the current validation loss is smaller than 0.996 of the best one, the early-stopping patience will increase by 0.3 times the current number of iterations;
- Training stops when the early-stopping patience is less than the current number of iterations.

For evaluation, five-fold cross-validation (CV) is performed throughout all experiments. Each fold is a combination of approximately 1/5 tracks of each

dataset. Every model is trained on four folds and cross-validated on the remaining fold, resulting in a total number of five validation scores, the average of which will be the final scores to be reported.

4.3 Results and Discussions

This study evaluates several variants of the proposed approach (as summarized in Table 4.1) in terms of the MIREX ACE standard metrics as well as the *ACQA*. All the metric in this section are already introduced in Section 2.3.1.

Dimension	Configurations
training scheme	completely random (CR); even chance (EC)
data size	JK; JKU; JKUR; JKURB

Table 4.1 Variants considered in this study.

4.3.1 On Notegram Feature

Sevenths, Inversions and ACQA — Table 4.2 shows the comparison between CR and EC training schemes on some uncommon (*non-majmin*) [BWF11] chords' *WCSR*s as well as the *ACQA*. The six chord types in the table are chosen because they have relatively more weights in pop/rock songs than the more long-tail ones such as *min/5* and *min/b3*. Note that *maj/5* and *maj/3* are also included in two other large vocabularies proposed by Mauch [Mau10] and Cho [Cho14].

	maj7	*7*	*min7*	*maj/5*	*maj/3*	*7/b7*	*ACQA*
CR	7.3	6.6	24.1	4.5	24.5	0.0	10.8
EC	14.6	9.9	30.9	12.0	32.4	7.8	13.2

Table 4.2 Comparison between CR and EC: seventh chords, inversions and ACQA scores; Dataset: JKURB; Feature: notegram

The results show that EC outscores CR in all categories, some of which by very large amount such as *maj/5* and *maj/3*. Although not all chord types' results are shown, the *ACQA* results suggest that the EC training scheme could lead to a much more balanced LVACE system under a skewed class distribution.

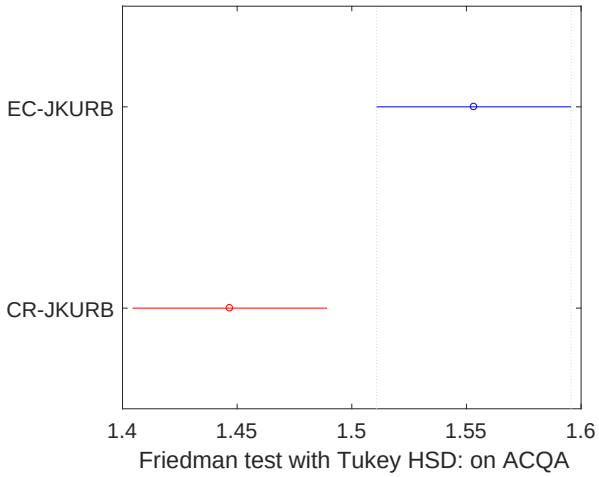

Figure 4.3 Multiple comparison test on ACQAs

We perform a Friedman test on the track-wise *ACQA* results of both systems. After that we use the Tukey HSD (honest significant difference) to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. The results as shown in Figure 4.3 confirm that EC is significantly better than CR in *ACQA*.

Major, Minor and WCSR — The EC trained system has a more balanced performance than the CR's, however, it scarifies common chords' *WCSR*s. Table 4.3 shows the comparison between CR and EC on some common (*majmin*) [BWF11] chords' *WCSR*s as well as on the overall *SeventhsBass WCSR*.

	maj	*min*	*WCSR*
CR	74.2	52.2	52.0
EC	67.8	51.4	50.6

Table 4.3 Comparison between CR and EC: major, minor and WCSR scores; Dataset: JKURB; Feature: notegram

Although the two schemes have very close scores on *min*, there is a large difference in *maj*. Due to the dominantly large weight of *maj* chords in the JKURB dataset combination, it eventually leads to CR's much higher *WCSR*, despite EC performs better in most of the other chord types. CR's much higher *maj WCSR* is not unexpected: since it draws each training case at random, the probability that each chord type gets "seen" by the neural net is subject to the distribution of chord types in the training dataset, and therefore the *maj* chords are "learned" much more than the other chords.

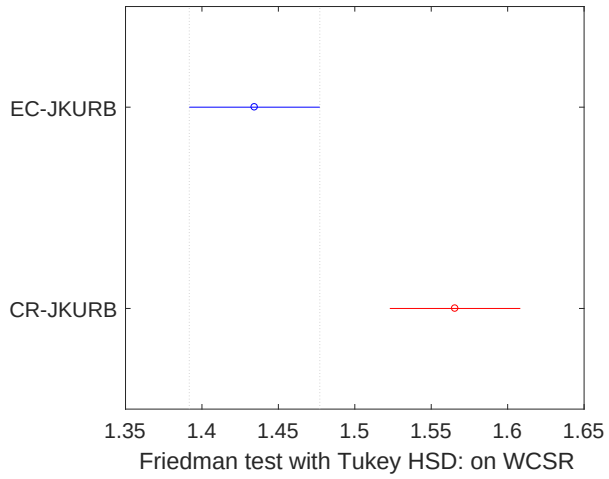

Figure 4.4 Multiple comparison test on WCSRs

We perform a Friedman test on the track-wise *WCSR* results of both systems. After that we use the Tukey HSD to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. The

results as shown in Figure 4.4 confirm that CR is significantly better than EC in *WCSR*.

On Different Datasets — For more convincing comparison results, the same experiment is run 4 times using different dataset combinations. Table 4.4 shows the results of JK, JKU, JKUR and JKURB. We only report the *WCSR* and *ACQA* for brevity.

	CR-WCSR	EC-WCSR	CR-ACQA	EC-ACQA
JK	46.4	46.4	13.5	15.5
JKU	50.4	49.1	11.2	13.5
JKUR	50.1	49.6	12.8	14.5
JKURB	52.0	50.6	10.8	13.2

Table 4.4 Comparison between CR and EC: WCSR and ACQA on different datasets; Feature: notegram

In all these experiments, the EC systems get higher *ACQA*s, but lower or equal *WCSR*s, than the CR systems. It is sufficient to say that EC is better at training a balanced performing LVACE system under skewed class distribution, while CR is better at training an LVACE system with higher overall performance.

For both training schemes, the increment of training data will lead to the increase of *WCSR*, but the same thing does not happen in *ACQA*. Assuming that every dataset contains a certain amount of noise (i.e., mis-labeled or mis-segmented chord regions), this observation could be tentatively explained as follows. *WCSR* is mostly relying on the quality of *majmin* chord labels, which are on average easier to be labeled. Therefore the increment of data will also increase the *WCSR* score. *ACQA*, however, is mostly relying on the quality of *non-majmin* chord labels, which are on average more difficult to be labeled. Therefore the increment of data could not guarantee the increase of *ACQA*

score, since it is hard to guarantee the proportion of *non-majmin* noise in the incremental data is smaller than those of the original data.

4.3.2 On Chromagram Feature

Overall Performances — Table 4.5 shows the overall *WCSR*s of systems implemented with both training schemes. Note that the *SeventhsBass* column represents the true evaluation score under the *SeventhsBass* vocabulary, while the other three columns to the left are scores with certain types of chord confusion tolerance.

System	*MajMin*	*MajMinBass*	*Sevenths*	***SeventhsBass***	*Segmentation Quality*
CR-JK	67.1	65.0	49.7	48.2	78.1
EC-JK	68.0	65.2	51.3	49.3	77.3
CR-JKU	67.4	65.6	54.9	53.4	76.9
EC-JKU	67.7	65.9	55.3	53.7	76.3
CR-JKUR	70.1	68.3	56.8	**55.2**	78.0
EC-JKUR	70.0	68.1	56.6	55.0	76.9
CR-JKURB	70.6	68.7	56.3	54.7	78.0
EC-JKURB	70.1	68.2	57.6	**56.0**	76.8

Table 4.5 WCSRs of different system variants; Feature: chromagram

Along the *SeventhsBass* column, there is a trend that for both training schemes, system performances increase with more training data. When the data set is fixed, the EC systems' *SeventhsBass* score are fairly comparable or slightly higher than the CR systems'. The same trend is also observed in columns of *MajMin*, *MajMinBass* and *Sevenths*.

For *segmentation quality*, CR systems are found to be consistently scoring slightly higher than the EC systems. Note that CR draws a training case from a random starting point which could be in the middle of a segment, while EC always draws a case from the start of a segment. We speculate this deviation

eventually leads to the CR systems' more exposure to segment boundaries during training, and thus they have a slightly better "understanding" of segments.

Balanced performances — Table 4.6 shows the systems' performances on some important uncommon chords' *WCSR*s and the *ACQA*. These three chord types (i.e. *maj*/5, *maj*/3 and 7/*b*7) are most commonly used in pop/rock songs among all uncommon chords in the *SeventhsBass* vocabulary. We could notice obvious performance advantages of EC in all these categories under the same amount of training data sizes.

System	*maj/5*	*maj/3*	*7/b7*	*ACQA*
CR-JK	17.9	43.2	28.9	16.6
EC-JK	23.3	42.7	**37.2**	19.7
CR-JKU	11.9	36.5	10.1	15.0
EC-JKU	20.3	40.2	35.4	19.2
CR-JKUR	18.8	46.0	7.0	16.6
EC-JKUR	**23.7**	**50.0**	32.4	**20.9**
CR-JKURB	11.9	38.5	4.9	14.7
EC-JKURB	18.7	42.2	24.0	18.4

Table 4.6 Some uncommon chords' WCSRs and the ACQAs; Feature: chromagram

For *ACQA*, again, we see that all EC systems have clear advantages over CR systems. This demonstrates that the EC training scheme can improve the vocabulary versatility of an ACE system.

It should be pointed out that in Table 4.6 neither the *WCSR* of a specific chord nor the *ACQA* is proportional to the amount of training data. Figure 4.5 and 4.6 show the *WCSR* trends of *maj/3* and *maj/5* chords. They all have local maximum at JK and JKUR and local minimum at JKU and JKURB.

Table 4.7 shows the distribution of common chords (majmin) and uncommon chords (sevenths and inversions) in the datasets we use. We see that the

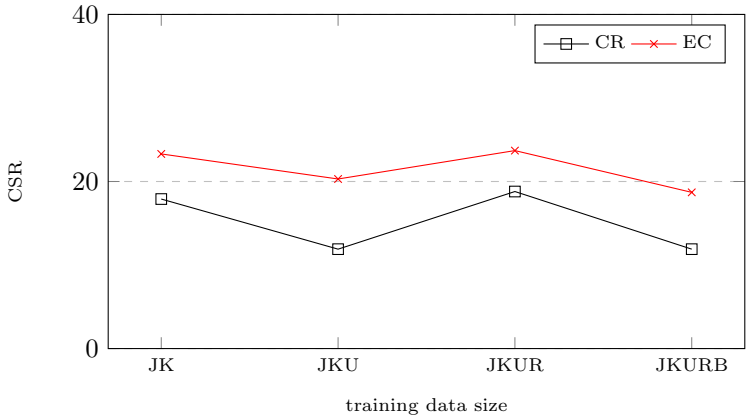

Figure 4.5 $maj/3$ performance v.s. training data size

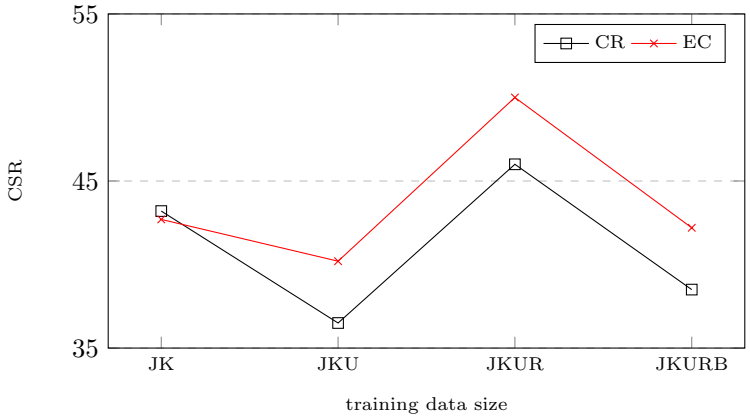

Figure 4.6 $maj/5$ performance v.s. training data size

datasets U and B have more uneven distributions of common and uncommon chords than dataset R, and therefore when they are mixed up with other data, the overall amount of uncommon chords' exposure will be decreased. This explains the CR's curve in Figure 4.6. For EC, each chord type has an even chance to be chosen as the start of a training case. Each training case has 500 frames, which contains multiple chords. Because of this, there is still uneven distribution of common and uncommon chords during the training process. The EC scheme in effect only boosts the exposure of uncommon chords to

Dataset	majmin%	sevenths%	inversions%
C	58.7	28.6	12.7
J	47.4	31.7	20.9
K	66.9	21.8	11.4
U	69.8	21.7	8.4
R	58.1	32.5	9.4
B	81.9	12.1	6.0

Table 4.7 Distribution of chords in the datasets

a certain level, but could not make the chances of common and uncommon chords totally even. Therefore, the same reason for CR also apply to explain EC's curve in Figure 4.6.

4.3.3 Baseline Comparison

SeventhsBass Evaluation — Finally, we compare the proposed LVACE approach with the baseline approach - Chordino[CMD+13]. It has a similar feature extraction module as the proposed approach and it supports a similar set of large vocabulary. Chordino is an LVACE system that represents the state-of-the-art in terms of the *ACQA*.

System	*MajMin*	*MajMinBass*	*Sevenths*	***SeventhsBass***	*Segmentation Quality*
CR-ch	70.6	68.7	56.3	54.7	78.0
EC-ch	70.1	68.2	**57.6**	**56.0**	76.8
Chordino	**72.4**	**69.1**	55.8	52.8	**83.8**

Table 4.8 Comparison between our systems and Chordino - overall performance; The systems are trained using JKURB

Table 4.8 shows the comparison, in which we select two systems trained with the largest possible amount of data with different training schemes. In terms of *WCSR*s, our systems slightly outperform Chordino in *Sevenths* and

SeventhsBass scores, but underperform Chordino in the more relaxed *MajMin* and *MajMinBass* scores.

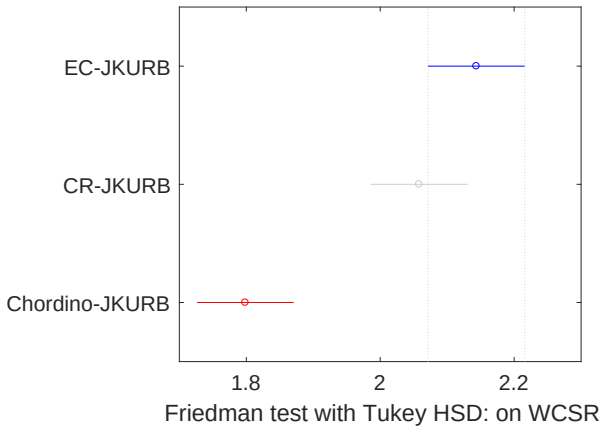

Figure 4.7 Multiple comparison test on SeventhsBass WCSRs

Of all the categories in Table 4.8, *SeventhsBass* represents the large vocabulary metric. We therefore perform a Friedman test on the track-wise SeventhsBass *WCSR* results of the three systems. After that we use the Tukey HSD (honest significant difference) to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.0. The results are shown in Figure 4.7, which confirm that both CR and EC are significantly better than Chordino in SeventhsBass *WCSR*, and there is no significant difference between CR and EC.

A large difference is found in *segmentation quality*, in which Chordino scores 5.3 and 7 points higher than our systems. As noted previously, the *segmentation quality* is indeed a problem in this approach. One could expect boosts of overall performance as well as balanced performance if the *segmentation quality* score could be raised to at least the Chordino's level.

Uncommon Chords and ACQA — Table 4.9 shows another set of comparisons

System	*maj/5*	*maj/3*	*7/b7*	*ACQA*
CR-ch	11.9	38.5	4.9	14.7
EC-ch	18.7	**42.2**	24.0	**18.4**
Chordino	**27.6**	29.8	**24.4**	**20.9**

Table 4.9 Comparison between our systems and Chordino - uncommon and balanced performance; The systems are trained using JKURB

in terms of uncommon chords' *WCSR*s and the *ACQA*. The CR system underperforms Chordino in all the categories except for *maj*/3. The EC system outperforms Chordino in *maj*/3 and is close to Chordino in both 7/*b*7 and *ACQA*.

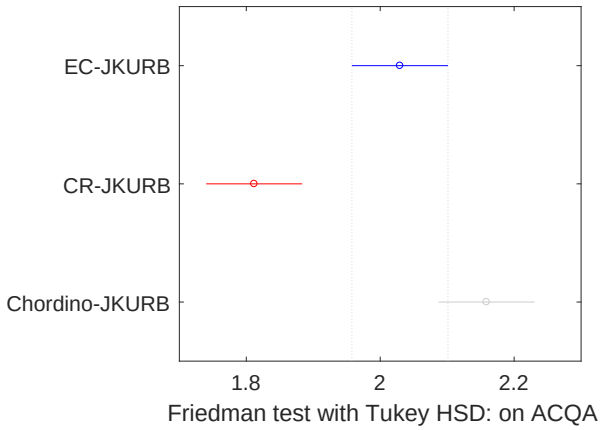

Figure 4.8 Multiple comparison test on ACQAs

We perform a Friedman test on the track-wise *ACQA* results of the three systems. After that we use the Tukey HSD to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. The results are shown in Figure 4.8, which indicate that both EC and Chordino

are significantly better than CR in *ACQA*, and there is no significant difference between EC and Chordino.

4.4 Summary

This chapter presents a BLSTM-RNN based LVACE system, trained using a skewed class oriented "even chance" scheme. Several system variants with different configurations are implemented and evaluated using both notegram and chromagram features as input.

According to the notegram's results, the EC training scheme is superior in both the uncommon (*non-majmin*) chords' *WCSR*s and the *ACQA*, at the expense of the common (*majmin*) chords' *WCSR*s and the overall *WCSR*.

According to the chromgram's results, the EC training scheme can improve uncommon chords' *WCSR*s as well as the *ACQA*, while still maintaining the overall *WCSR* performance at the level of the CR training scheme.

When compared with Chordino, the EC system's performance is still very competitive since it significantly outperforms Chordino in *SeventhsBass WCSR*, and it does not significantly differ from Chordino in *ACQA*. However, the EC systems suffer from bad *segmentation quality*, which should be well noted as the biggest drawback of the proposed LVACE approach. Improving the *segmentation quality* may require a more focused learning strategy towards chord segmentation.

A successful LVACE system is marked by both high *WCSR* and high *ACQA*, because human chord recognition experts are able to achieve both of them. The EC training scheme is a technique to improve a system's *ACQA*,

thus it is a valuable approach to consider when we design an LVACE system in the future.

Chapter 5

A Preliminary Approach to Automate Jazz Chord-Scale Estimation and Jazz Improvisation

Previous chapters have explored ACE in the *SeventhsBass* vocabulary. This vocabulary is mostly covered in music genres such as pop, rock, and folk songs. But chord types such as sevenths extensions, suspensions and alterations are not included in *SeventhsBass*. This chapter, following the original large vocabulary spirit of Fujishima [Fuj99] and Sheh and Ellis [SE03], tries a preliminary ACE solution that handles a much larger vocabulary, and beyond that, puts together a "chord-scale" estimation system that could be deployed in an automated jazz improvisation platform.

5.1 Jazz Fundamentals

This chapter focuses on a chord vocabulary of triads, sixths, sevenths and their extensions, suspensions, and alterations. In particular, it targets at jazz, which is an improvisation based style with complex harmonic structures [HM13]:

> One thing that distinguishes mainstream jazz harmony from other tonal styles is the tremendous amount of harmonic color that arises due to the pervasive use of tertian extensions of the basic chord types.

Different from pop, rock and folk music, which usually have long chord progressions within the same key and only modulate at most a few times during a piece, jazz music often contains a lot of key modulations. A *modulation* is [Ran99]:

> in tonal music, the process of changing from one key to another, or the result of such change.

Musical key and chord progression are closely related to each other. Many MIR systems [CML07, NS09, MD10b, PT12] have been attempted to extract both at the same time. But these works are all experimented under low modulation rate contexts. While under a jazz context, where modulation rate is much higher, these classical approaches may or may not apply. Hence besides ACE, this chapter will investigate on a different key tracking algorithm in order to also capture the modulations. It should be noted that in *modal jazz*, where a piece is constructed horizontally around the melody instead of vertically around the harmonics, the functional role of "chord progression" is weaken or eliminated, but there are still harmonic movements, regardless of keys or chords, to support the melody lines.

As jazz is an improvisation based music style, extracting the chord progressions and key modulations could help determine/imply the *chord-scale* candidates to improvise over a given harmonic segment. A *chord-scale* is [HM13]:

> a linear rendering of a complex chord - an extended chord structure, with tensions and non-chord tones arrayed within an octave.

Each chord-scale is constructed by a root and a scale, where the root is similar to the root of a chord. There are seven commonly used scales in jazz that are derived from the *major* scale. They are called *mode*s or *modal scale*s. They are:

1. Ionian: $WWHWWWH$
2. Dorian: $WHWWWHW$
3. Phrygian: $HWWWHWW$
4. Lydian: $WWWHWWH$
5. Mixolydian: $WWHWWHW$
6. Aeolian: $WHWWHWW$
7. Locrian: $HWWHWWW$

where W is a whole step, and H is a half step (note that $W = 2 * H$). The sequences indicate the arrangement of whole/half step intervals within the scales. These modes, from 1 to 7, can be memorized as left-circular shifting the *major* scale ($WWHWWWH$) one note at a time. Likewise, modes can also be built based on the *harmonic minor scale* ($WHWWHW^{+}H$, where $+$ means an additional half step) or the *melodic minor scale* ($WHWWWWH$). Alternatively, there could be other scales out of this construction methodology, such as the *whole-half diminished scale* ($WHWHWHWH$), *half-whole diminished scale* ($HWHWHWHW$), *whole-tone scale* ($WWWWWW$) or *chromatic scale* ($HHHHHHHHHHHH$).

Chord	Scale
7	Mixolydian, Phrygian-Dominant, Whole-tone
maj7	Lydian, Lydian-Dominant, Ionian, Ionian#5
min7	Dorian, Phrygian, Aeolian
min7b5	Locrian, Locrian#2
7b9	Phrygian-Dominant
maj7#11	Lydian
maj7#5	Ionian#5
dim7	Whole-half Diminished

Table 5.1 Chord-scale choices examples

Normally, a chord-scale is chosen based on the current musical key context, which is determined by the underlying *harmonic functional group* [HM13, Lev11]. Table 5.1 shows some choices suggested by a jazz guitar tutorial book [Sch03]. For a specific chord type, there could be multiple choices of scales, but the best picks will be determined by the musical key context, where the scale of choice cannot strongly indicate otherwise.

Chord-scale system is a good way to analyze jazz harmony and melody. But in real improvisation, jazz musicians seldom think of this system. Instead, they memorize all these by heart and improvise using the variations of the passages and patterns they learn through extensive exercises [Few10]. To be precise, instead of generating notes from the scales, they generate phrases that fit the context. Infinite number of note sequences can be created, but those phrases, within all these sequences, are among the most acceptable ones to human musical aesthetics. In addition to creating phrases within a single harmonic region, jazz musicians also pay much attention to the coherence of nearby phrases so as to make the best musical sense possible.

5.2 Automatic Jazz Chord Estimation

To perform ACE for jazz, the system framework in Chapter 3 is used. Assuming the segmentation pass can achieve high output quality, the remaining task is to classify chords independently in each segment. Figure 5.1 is a succinct version of the system framework, where the segment tiling process is absorbed into the chord classifier module.

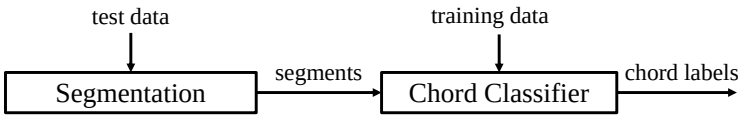

Figure 5.1 The Jazz ACE system. It is equivalent to the system framework in Figure 3.1

Table 5.2 shows all ACE configurations considered in the experiment. They are chosen based on the results in Chapter 3.

Dimension	Configuration
deep neural nets	FCNN, DBN, BLSTM-RNN
hidden layers	(800,800)
segment tiling	6
feature level	notegram(-ns), chromagram (-ch)

Table 5.2 ACE configurations

5.2.1 Experimental Setup

In this study, 99 pieces of jazz chord comping + soloing extracted from a jazz guitar book [Sch03] (JazzGuitar99) are used as the training dataset, and 7 pieces of jazz comping from Gary Burton's on-line course [Bur] (GaryBurton7) are used as the test dataset. JazzGuitar99's annotations are taken directly from the book, and GaryBurton7's annotations are taken from the leadsheets

provided along with the course. All training data are augmented 12 times using the same procedure as in Chapter 3 and 4. The vocabulary has 36 types [1], which is much larger than the *SeventhsBass*. Note that inversions are not considered in this preliminary jazz ACE study because: 1. there are very few inversion notations in the currently used datasets; 2. it will result in a huge number of classes.

5.2.2 Results and Discussions

Following the MIREX ACE convention, system performances are reported using *WCSR*. The *WCSR* computing procedure in its fairest/strictest sense should regard each chord as it is without applying any sort of mapping scheme. In the following, each system is evaluated in this way. The baseline is an augmented Chordino with jazz vocabulary extension (Jazz-Chordino). The augmentation is done within its GMM-HMM process by applying the jazz chord dictionary to the Gaussian emission model, whose setting is given in Table 5.3.

	μ	σ^2
Bass - Chord Bass	1	0.1
Treble - Chord Note	1	0.2
Neither bass nor treble	0	0.2
N.C. (for all notes)	1	0.2

Table 5.3 Gaussian model of Jazz-Chordino

[1] They are: *maj*, *min*, *min*6, 6, *maj*7, *maj*7#5, *maj*7#11, *maj*7*b*5, *min*7, *minmaj*7, *min*7*b*5, *min*7#5, 7, 7*b*5, 7*b*9, 7#9, 7#5#9, 7#5*b*9, 7*b*5*b*9, 7#5, 7*sus*4, *aug*7, *dim*7, *maj*9, *min*9, 9, 9#11, *min*11, *min*11*b*5, 11, *min*13, *maj*13, 13, 13*b*9, 69 and *N.C.*

systems	*WCSR*	*SQ*
Jazz-Chordino	57.99	81.68
Jazz-FCNN-ns	61.81	76.18
Jazz-DBN-ns	62.33	80.73
Jazz-BLSTM-ns	**66.41**	80.78
Jazz-FCNN-ch	**65.65**	81.94
Jazz-DBN-ch	4.56	20.42
Jazz-BLSTM-ch	63.72	82.22

Table 5.4 Jazz ACE performances. The Jazz-DBN-ch result, due to excessive regularization, could be considered as an outlier.

All systems are tested using the GaryBurton7 dataset [2]. Results are shown in Table 5.4. Jazz-BLSTM system performs the best, and outperforms Jazz-Chordino by about 10 points. The ranking is very similar to the *SeventhsBass*', but these results are in a sense more convincing, since the test set is not dominated by *major* and *minor* chords. In fact, the chord composition in GaryBurton7 is relatively balanced, although rare chords are still rare. These results demonstrate the advantage of the proposed system framework for very large chord vocabulary.

Meanwhile, notice that the *SQ* (segmentation quality) of these systems are all relatively high, and these are achieved in pure jazz test audio. All systems share Jazz-Chordino's GMM-HMM segmentation process. The differences among the *SQ* scores are caused by different merging of consecutive chords in different systems. (In this sense, the Jazz-DBN-ch system is probably constantly generating the same chord).

[2]Composition of chords in GaryBurton7: *maj*:0.09; *min7*:0.13; *7*:0.22; *min7b5*:0.12; *7b9*:0.06; *min*:0.1; *maj*:0.14; *others*:0.14.

5.2.3 Extension - Jazz Scale Estimation

The above jazz ACE system can be extended to jazz scale estimation, thereby achieving a unified chord-scale estimation system. As reviewed in Section 5.1, the scales are chosen based on the harmonic functional group, which establishes a temporary tonal center, or a key. Consequently, a natural way to perform jazz scale estimation is by local key estimation. Figure 5.2 shows

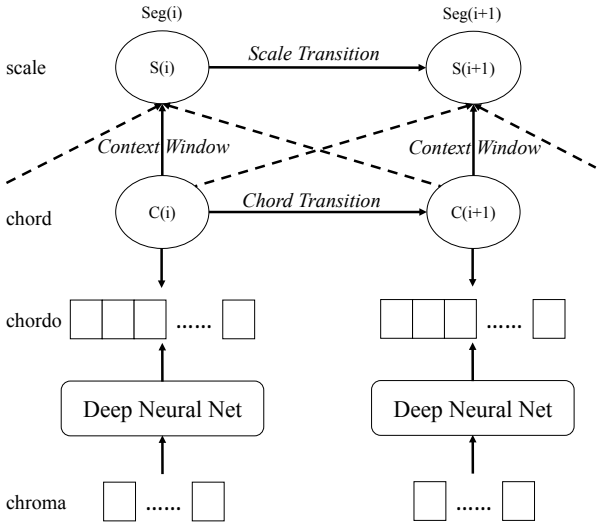

Figure 5.2 Template-based local scale tracking. A scale is estimated within a context window of chords. "S" stands for segment; "C" stands for chord.

the extended ACE system for local scale estimation, where "chordo" means the probabilities of each chord conditioned on the chroma input. Different from the chord-key estimation systems proposed by Mauch [MD10b] and Catteau [CML07], the scale is not estimated by a generative model, but a discriminative template-based model similar to Rocher et al.'s approach [RRH+10]. Instead of using a well-known key profile [Tem04], binary "scale templates" (ST) are used to compute a local scale posterior probability surface from the "chord templates" (CT), similar to the key estimation method in [HDZ+15]. Since

each local scale cannot be determined by just one chord, a context window is introduced to collect a neighborhood of 3 chords. As a result, a scale's fitness of the context is computed as:

$$\text{fitness} = \sum_{i=1}^{12} \sum_{j=1}^{3} ST_i \cdot CT_{ij}, \tag{5.1}$$

For a given context, the system will decide a list of scale candidates, and then pick the one with the best fitness:

$$scale = \max_{k}(\text{fitness}_k). \tag{5.2}$$

The template-based local scale estimation is implemented during the ISMIR 2016 Hackathon [3]. Figure 5.3 shows an example output of this chord-scale estimation process with "Olhos de Gato", where the labels above the stave indicate the estimated chord-scales.

However, since this is a relatively new direction in MIR, there is not any well-established labeled dataset available for evaluation. Therefore this system has not been formally evaluated under any objective measure yet.

The jazz chord-scale estimation research has its straightforward application to new interfaces for musical expressions. In the following, two mobile jazz improvisation platforms (WIJAM and ArmKeyBoard) will be introduced in Section 5.3 and 5.4. Finally, Section 5.5 will seek to put the chord-scale estimation system in a fully automatic jazz improvisation context.

[3] `http://labrosa.ee.columbia.edu/hamr_ismir2016/`

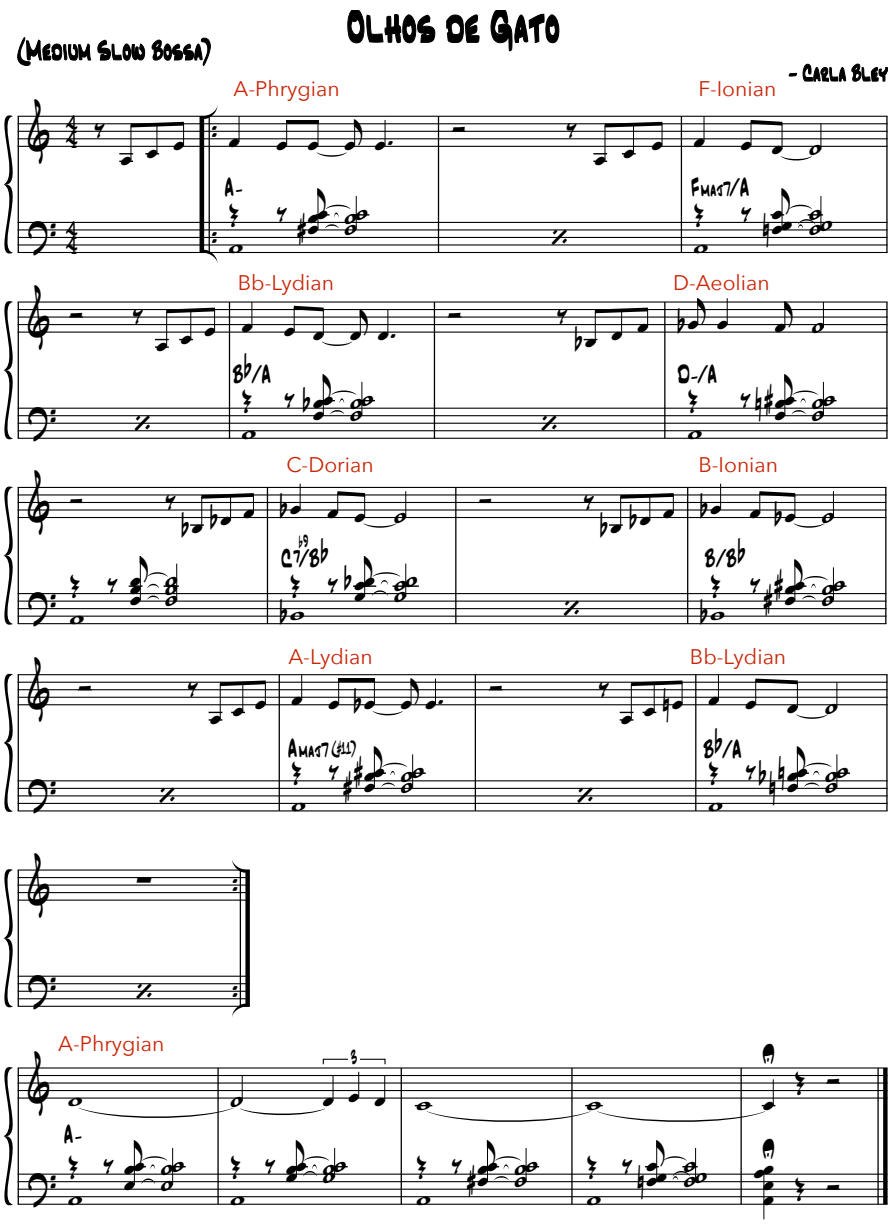

Figure 5.3 Chord-scale tracking demo output from a leadsheet

5.3 Jazz Improvisation Platform - WIJAM

WIJAM is an impromptu iOS mobile application for a group of musical novices to jam along with a music master. The master provides the backings, directs the musical flow and gives feedbacks to the players. The players improvise along with the master's guidance.

WIJAM, according to Weinberg [Wei05], is a "small-scale local system", which can be characterized as a "collaborative musical network ... that support three to ten players in close proximity, which allows for detailed and subtle interpersonal interactions". Under the "theoretical framework of musical interconnectivity" [Wei05], WIJAM is structure-centered and process-centered, where the "structure" is in the absolute musical arrangement control of the WIJAM master, and the "process" is referring to the WIJAM players' expressing their musical feelings within the bounds defined by the WIJAM master.

In terms of organization and architecture, WIJAM is a "synchronous centralized network" with a "flower topology" [Wei05], where music is being generated in real-time with a central hub, and the degrees of freedom in terms of the nodes' expression are limited to a level such that the musical outcomes are at least harmonious.

5.3.1 Design Concepts

WIJAM's design follows Blaine and Fels' guidance [BF03b]:

> The underlying premise of most collaborative interface design is that with various design constraints, playing music can be made

> accessible to non-musicians ... at the expense of limiting the musical range and possible gestures associated with sound in a collective space.

But as pointed out in the same paper, such a guideline has its drawback:

> ... many of the simple-to-use computer interfaces proposed for musical control seem, after even a brief period of use, to have a toy-like character and do not invite continued musical evolution.

Although the authors argue that this is only true for expert musicians, "balancing this trade-off is a key concern for designers" [BF03a]. This point is echoed in many other papers [XLDJ11], emphasizing that "provide novices with essential goals and experts with additional goals". Trying to meet this balance, WIJAM equips the players with an easy to learn and easy to play keyboard, while at the same time providing "a knowledgeable person to stand by and assist the players" [BF03b]–the master. The players performance bound is imposed upon by the master, which should give a large enough space for the expert players to express their virtuosity.

5.3.2 System Overview

Figure 5.4 shows the overview of WIJAM. There are two types of active entities: the master, who initiates and orchestrates the jamming, and the players, who participate in the jamming. There is a passive entity: a loudspeaker system, which is connected to the master.

Figure 5.5 is the flowchart of a WIJAM session. At the beginning, the master hosts a *MIDINetworkSession* as a jam session service to be discovered and connected to. Then the players join the session. The master sends an

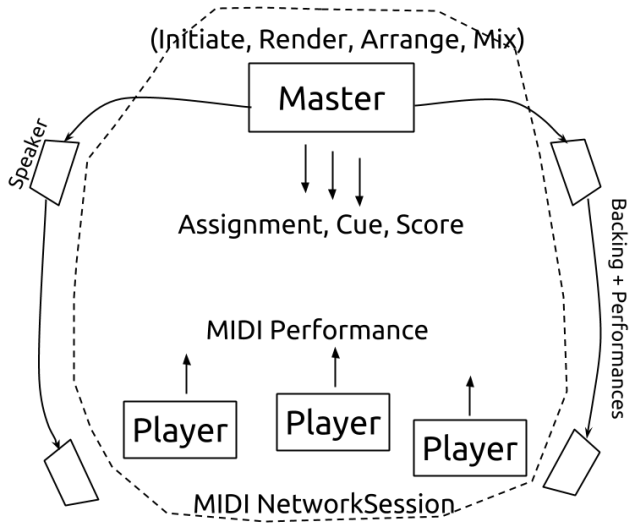

Figure 5.4 The overview of WIJAM. The music collaboration is achieved under the "master-players" paradigm.

"assignment" package to each player. The package contains an instrument ID, and a chord-scale. The master signals the start of jamming by switching on the backing music. The players jam by sending MIDI performances to the master via *coreMIDI*. The master monitors all the performances, renders the notes in the virtual instruments and mixes them with the backing music into one output stream. While jamming, the master orchestrates everything to fit to the backing music by instantly changing the assignments or providing performance feedbacks to the players such as "good", "solo", "fast", "calm", etc. When they finished jamming, the master gives an overall score as well as an individual score to each player.

5.3.3 Master Machine

The internal structure of the master machine [4] is shown in the middle part of Figure 5.7. The core modules are described as below:

[4] `https://github.com/tangkk/MasterMachine.git`

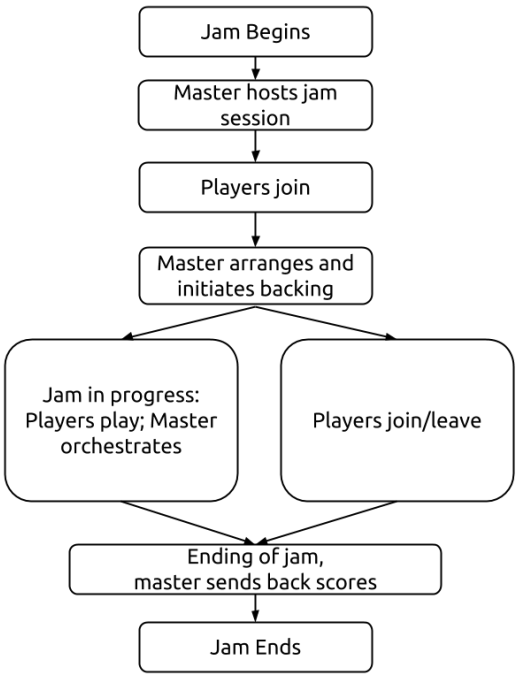

Figure 5.5 The flowchart of a WIJAM session.

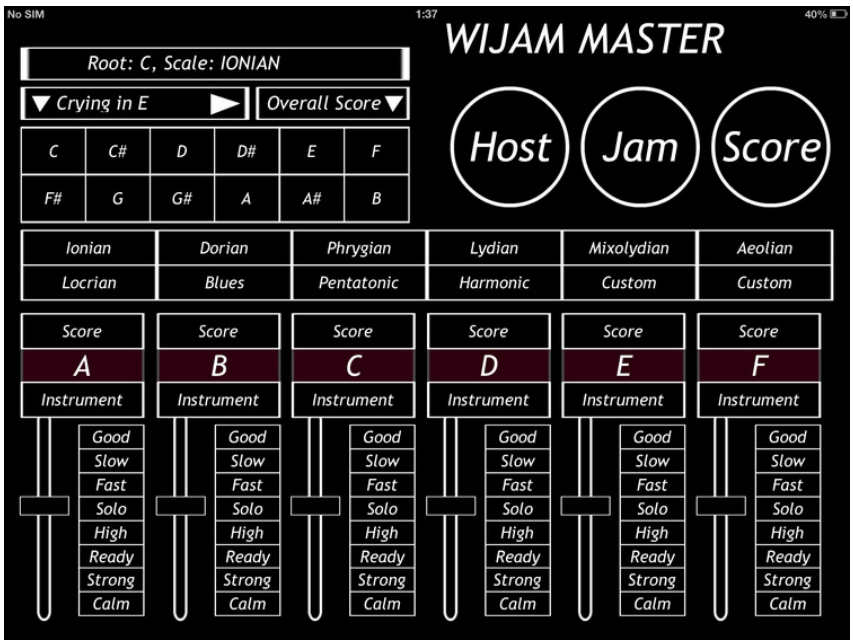

Figure 5.6 WIJAM's Master Interface

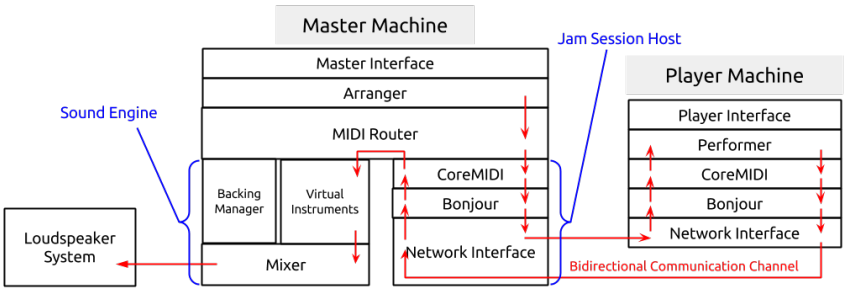

Figure 5.7 The internal layering and structure of WIJAM. The data flows between the master machine (middle) to the player machine (right) through the bidirectional communication channel.

Master Interface — The master interface is shown in Figure 5.6. At the top left corner there is a backing selector which allows the "master" to initiate a piece of backing music. Moving downwards there is a chord-scale panel. They are particularly important for orchestrating the musical flow. At the bottom there are six consoles, from A to F, each of which is assigned to a player. These consoles are used together as a mixer.

Arranger — This layer allows the master to manipulate chords, scales, instruments, and feedbacks. These messages are encoded via *MIDI sysEx* messages.

Virtual Instruments — It stores a library of digital musical instruments as samplers. The audio samples are managed using *AUPreset* and the *AUSampler* technologies. The *AUPreset* files are used to map audio samples directly to MIDI numbers and velocity ranges.

Jam Session Host — It hosts a service which can be discovered and connected to by the player machines. The *MIDINetworkSession* is used to enable a *Bonjour* service. It is to be discovered by the player machines' *Bonjour* service

browsers, and to be connected to by the player machines' *MIDINetworkSessions*.

5.3.4 Player Machine

The player machine's layering is described on the right hand side of Figure 5.7 and its interface on Figure 5.8 with the jam console on the left and the keyboard on the right. [5] The jam console is for discovering and connecting to the master machine, and the keyboard is for musical expression.

Figure 5.8 WIJAM's Player Interface

The following items summarize the working mechanism of the keyboard:

- The y position determines the pitch, where higher position yields higher pitch.
- The keyboard is filled with the chord-scale assigned by the master machine.
- The x position determines the note velocity. A larger x position value gives higher velocity;

[5] `https://github.com/tangkk/PlayerMachine.git`

- When sketching on the keyboard, a note is generated when the it starts or when the curve comes to a stationary point.

This keyboard is easy to use for novice players. By abstracting away most of the complicated musical context, it allows the players to do a high-level melody improvisation.

5.3.5 Demos

There are two basic demos [6] and one advanced demo [7] for WIJAM. The basic demo shows how WIJAM works, and provides tips for using the master machine and the player machine. The audio track is an unmodified recording of the original playback. Before the jam starts, the players are told to follow their "musical feelings" based on the backing. As can be heard in the video, the music outcome is quite pleasing, even at the critical "key modulation" points, i.e., 2:30 and 2:43. Note that as the jam goes on, because the players have little idea about what the exact notes they are playing, the "avoid note" (the most dissonant note within the chord-scale) might also be played on downbeats. This may happen, but with a very small chance.

The "trombone" player and the two "guitar" players are with zero training in musical instruments. The "piano" player has some limited experience with piano before. Even so, the piano player is able to perform a very beautiful solo during the jam. The "piano" player apparently plays the best music in this jam session. This somewhat justifies the "balance" problem introduced earlier in this section, that WIJAM actually allows advanced players to demonstrate

[6] http://www.youtube.com/watch?v=Y0PnKBrgzgw
[7] http://www.youtube.com/watch?v=16dWj5G9UKw

their musicianship while also enables novice players to perform at an acceptable level.

The advanced demo is a bonus track featuring the author's playing and overdubbing a whole jam session. It shows some advanced features of the PlayerMachine. The audio track is an overdubbing of each instrument track plus the backing track.

5.4 Jazz Improvisation Platform - ArmKeyBoard

The piano keyboard, although is versatile and popular, has a lot of drawbacks. The same type of chord or scale in different keys are laid out differently, which adds to the learning difficulty. Additionally, the keyboard has a linear layout of the black and white keys, which works well with music expressions that exhibit certain linearity, but is less effective for modernistic non-linear styles such as that of serialistic and stochastic music [Uch].

Trying to solve the above problems, a new type of keyboard is designed. Based on the NIME design principles [Coo01], specifically the "Make a piece, not an instrument or controller" and the "Instant music, subtlety later", the keyboard leverages a chord-octave-scale sequence grid to pack 88 keys into a 15–17 keys-sized screen, and features an almost zero learning curve for the production of beautiful and sophisticated melodies. It offers both linear and non-linear layouts. The non-linear layout is mapped to a user chosen image by an algorithm based on contour separation and tonal hierarchy. This is called "ArmKeyBoard", where "ArmKey" means suitable, comfortable, and in-tune, in the author's spoken dialect.

5.4.1 Two Types of Keyboard Layout

In the remaining discussion, "keyboard" refers to an instrument implementing a series of key-note pairs and deterministically generates a note when a key is pressed. "Layout" refers to the spatial arrangement of those key-note pairs.

Linear layout is the characteristic of a piano. From left to right, each key is mapped to a unique note value (in MIDI's terms). Every next key is mapped to a note value exactly 1 higher than the previous one. This has a significant impact on music making. Because people naturally feel more comfortable with playing on adjacent keys than non-adjacent keys, it leads to smaller intervals' appearing more often than larger ones, as can be seen in music literature such as *the Real Book* [Mus04].

Non-linear layout has many more possibilities, such as a random note being paired with a random key or one note being paired with several keys. Pay attention that in the current discussion, several notes being paired with one key is not valid by the definition of keyboard. Non-linearity may further allow using any key setting other than the traditional setting. For example an image can be divided into several regions, each serving as a key of the keyboard. The idea of non-linear keyboard is not new. There are existing applications [KON] or papers [KW11] talking about similar ideas, most of which are built around the idea of sampling. In ArmKeyBoard, however, the audio content generated by a key is a note.

5.4.2 Chord, Scale and Octave

ArmKeyBoard treats the small screen as a cache, caching the currently playing chord-scale in the current octave range, while other octaves and chord-scales are waiting to be loaded when needed. In the current design, 15–17 notes, i.e. two octaves of a scale, are cached.

Changing chord-scales or octaves on a piano in real time is easy for a pianist, but could be a nightmare for non-pianists. Therefore, ArmKeyBoard needs a special mechanism to load other chord-scales and octaves into the foreground, so that the player can easily switch music expression ranges in real time. To this end, a chord-octave-scale sequence grid as shown in Figure 5.9 is designed.

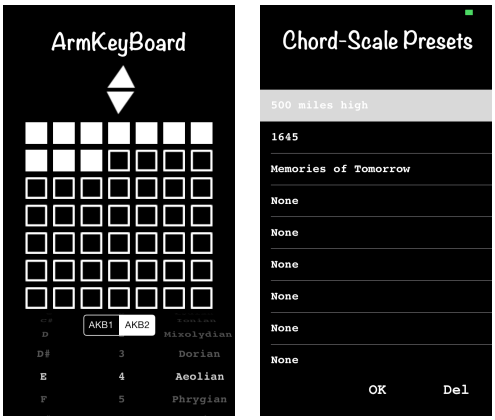

Figure 5.9 The left Figure shows the chord-octave-scale grid, where each square is a chord-octave-scale combination (such as C-4-Lydian), and consecutive squares form a sequence; The sequence is read from left to right, and when it reaches the rightmost square, it goes back to the leftmost square on the next line; The right Figure is the chord-octave-scale preset browser, showing the already saved chord-octave-scale sequence presets.

Users can switch between different chord-octave-scales using the gravity X gesture (Figure 5.10). Meanwhile, users can also change octaves within the

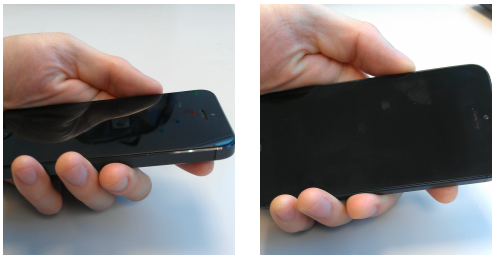

Figure 5.10 Gravity X gesture, which is used for switching to the next or previous page of notes.

same chord-scale using the swipe gestures. To summarize, the screen is a cache of the active note space, while the spaces around the active space can be loaded in real-time via the gravity X or swipe gestures, as indicated in Figure 5.11.

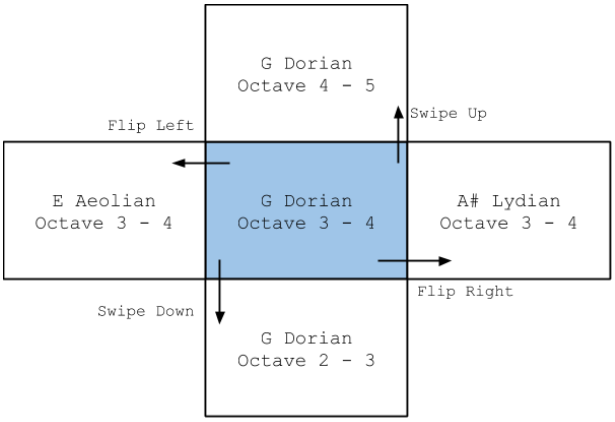

Figure 5.11 Chord-octave-scale control. The horizontal arrows indicate changing page of notes according to chord-octave-scale sequence, and the vertical arrows indicate changing page of notes to a higher or lower octave.

5.4.3 Expression Parameters

Because the proposed design is based on the piano keyboard, the most dominant expression parameter, velocity, should be implemented. In the linear layout, since key-note mapping is 1-to-1 and the position of each key is equally distributed along the y-axis, velocity can be easily controlled by position X.

While in the non-linear layout, position X cannot be used because keys can be in any shape and at anywhere; thus the Gravity Y is used to control the velocity (Figure 5.12). Besides, the gravity Z gesture is used to restart the keyboard again (Figure 5.12).

Figure 5.12 Gravity Y gesture (on the left) is used to control note velocity: a smaller velocity corresponds to a larger angle to the horizontal plane; Gravity Z gesture (on the right) is used to restart the keyboard

5.4.4 Linear Layout and Mapping

ArmKeyBoard has both linear and non-linear layouts. They are called "AKB1" and "AKB2" respectively. The implementations are based on the iOS platform.

"AKB1" (Figure 5.13) contains 15–17 notes within the active chord-octave-scale and they are mapped linearly to 15–17 bars equally divided along the y-axis. The velocity is controlled by the X position.

5.4.5 Non-linear Layout and Mapping

"AKB2" is a user selected image (Figure 5.13). The image is algorithmically divided into contours and they are mapped to the 15–17 notes within the currently active chord-octave-scale. The algorithms are described as follows.

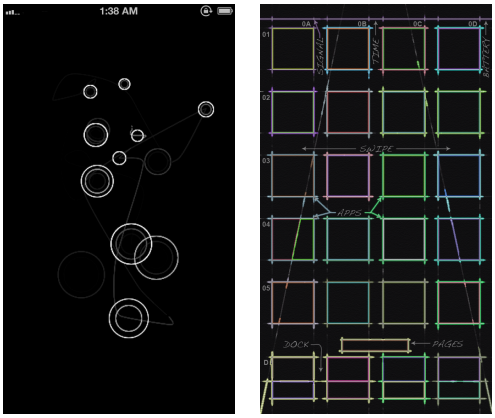

Figure 5.13 AKB1 (on the left) is a linear keyboard with a higher pitch at smaller y position, and larger note velocity at larger x position; AKB2 (on the right) is a non-linear keyboard mapping a page of 15–17 notes to the contours.

Contour Separation — The contour separation is processed using OpenCV [Ope]. The image is first transformed to an OpenCV *Mat*, which is then passed to a contour separation function. The function performs the following five steps: step 1, turns the *Mat* into gray scale and slightly performs a blur operation on it; step 2, passes the output of step 1 (a gray scale *Mat*) to an edge detection function; step 3, passes the output of step 2 to a *findContour* function, which finds contours, stores them in an array and calculates the contour hierarchy (a tree structure describing the inclusion relationship of contours); step 4, calculates the area of each contour, discards those below a certain size and deletes their nodes in the hierarchy; step 5, creates an outer contour which is the whole screen subtracted by all the contours output by step 4. The final output is an array of valid contours (each contour is itself an array of its vertices), and a hierarchy structure describing the inclusion relationship of these contours.

Contour Ranking — The next step is to decide the importance of each contour. The minimal musical concern is, when the keyboard is played, the notes being generated should at least imply the currently active chord-scale most of the time. Note that it is not necessary that it should "always" behave this way. For example, in *G-Ionian*, the keyboard is supposed to generate notes that form a tonal gravity at G and imply the G major chord most of the time, but sometimes it may also sound like *C-Lydian* (tonic at C).

More assumptions are needed to connect this musical concern to contours. Assume that most users tend to tap on: 1, a contour with a larger area; 2, a contour closer to the center of the screen; 3, a contour that contains more sub-contours. Based on how often most users will tap on a contour, its importance can be determined. Thus in the implementation, contours are ranked based on the weighted sum of the above three indexes. This corresponds to how important a note is in implying a certain chord-scale, which will be discussed below.

Tonal Hierarchy — Similar to ranking contours, we could rank the importance of notes. According to [Jar95], there is a certain tonal hierarchy within a chord-scale being played in bebop style jazz music, and this actually corresponds to the avoid note issue [NU87]. The tonal hierarchy theory says during the performance of a certain chord-scale, some notes are more often played than others. If the notes are to be divided into a hierarchy according to how often they appear, the first class contains chord tones (or chord notes), the second class contains those a whole step above the chord notes and finally those half step above, with exceptions.

Scale	L1	L2	L3
Lydian	1, 5, 3, 7	2, #4, 6	
Ionian	1, 5, 3, 7	2, 6	4
Mixolydian	1, 5, 3, b7	2, 6	4
Dorian	1, 5, b3, b7	2, 4	6
Aeolian	1, 5, b3, b7	2, 4	b6
Phrygian	1, 5, b3, b7	4	b2, b6
Locrian	1, b5, b3, b7	4, b6	b2
Lydian b7	1, 5, 3, b7	2, #4, 6	
Altered	1, 3, b7	#4, b6, b2, #2	5
Whole-half Diminished	none	none	
Melodic Minor	1, 5, b3, 7	2, 4, 6	

Table 5.5 Tonal hierarchies in ArmKeyBoard. L1 is the first level of notes which are to be mapped to regions with the highest importance, and L2 is to be mapped to regions with the second highest scores, then L3 to be mapped to the least important regions.

Table 5.5 is a result of the tonal hierarchy theory, which lists all the hierarchies [NU87] of some of the most frequently used chord-scales [Bur]. The scale degree notation is used instead of note name. The diminished scale has no hierarchy in this taxonomy.

The Final Mapping — The final mapping is not so obvious as it may seem. Although there are already a ranking of contours and a ranking of 15–17 notes within a chord-scale, they are by no means simply 1-to-1 mappings, because in reality it is not clear how many contours there are and how large each of them is until the user selects an image. In view of this complication, a heuristic based algorithm is devised to do this final mapping:

Algorithm 3 Contour-Note mapping

1. Divide the screen size by the number of notes, and name the result as RPN.
2. Look at the $ratio = Area(contour)/RPN$ of the top item of the sorted contour list (regarded as a stack). If $ratio >= 1$, do step 3; otherwise do step 4. Repeat until there is no contour left in the stack.
3. Map notes to contour: pop the contour, pop the top $ceil(ratio)$ notes and pair them up. Go back to step 2.
4. Map contours to note: pop the contour, pair it up with the first note. Add $ratio$ to $accum$. If $accum >= 1$, clear $accum$ and pop the note. Go back to step 2.

With this, the most important notes are mapped to the most important contours, and contours with larger areas will contain more notes. But since multiple notes cannot be mapped to a single key, they need to be decoupled within a contour that has more than one note. Instead of further separating a shape-unpredictable contour into several sub-contours, the notes are distributed across the contour according to a formula related to the contour's pixels and their RGB value:

$$noteIdx = ((X + Y)\%10 + (R + G + B))\%(15 \ or \ 17), \tag{5.3}$$

where 15 or 17 is the number of notes. This heuristic tries to make position affects less and RGB affects more, while making sure all the notes are included regardless of the shape of the contour.

5.4.6 Demos

There is a demo video of ArmKeyBoard [8]. The first part shows ArmKeyBoard's performance on a jazz backing track, and the second part shows a few solo performances. Besides the demo, the author has also tried the ArmKeyBoard in Gary Burton's on-line jazz improvisation course [Bur] to complete the peer reviewed assignments. In a total number of 6 assignments, it gets an average of 3 points out of the maximum 5 points.

5.5 Going Further

The modules in Section 5.2 to 5.4 can be put together and form a semi-automatic or fully automatic jazz improvisation machine. The differences among these approaches are the way they deal with note generation.

5.5.1 Chord-scale informed user improvisation

The simplest form would be to only provide the improvisation platform with a timed chord-scale sequence, and let the players to input note sequences. Firstly, the jazz backing track is ACEed into a segmented chord-scale sequence, then it is programmed into the improvisation platform. The information can automatically guide users with little musical knowledge to improvise jazz with at least proper choices of chord-scale.

5.5.2 Markov model note sequence generation

A more intelligent approach is to generate a note sequence within a chord-scale context using a Markov model. This model has two sets of parameters:

[8] `https://www.youtube.com/watch?v=ZhTleEXKeu4`

the note prior probability matrix and the note transition matrix. These matrices can be trained from existing jazz solo datasets (in MIDI) such as the Weimar Jazz Dataset [9][AFPZ13].

There is a preliminary implementation of this Markov model based note generator together with the template-based scale tracker during the "Science of Music Hackathon" in August 2016 [10]. In this implementation (Figure 5.14), a simplified version of the model is built to generate a sequence of "pitches" instead of "notes", where the former do not have durations but the latter do. The Markov note sequence generation is constraint by the chord-scale context. Only pitches in the chord-scale are allowed.

Figure 5.14 The Markov model for pitch sequence generation

The code base of this project is available on-line [11]. Several preliminary demos can also be found in the code base, showcasing several simple chord-melody outputs from the system.

5.5.3 RNN-DBN automatic note sequence generation

To the other extreme, it could be a fully artistically automatic improvisation process if the "patterns of choices" from real jazz practice [Few10, Whi12] are modeled and learned. These are sequences of notes within the context of

[9] http://jazzomat.hfm-weimar.de/dbformat/dboverview.html
[10] http://labrosa.ee.columbia.edu/hamr_ismir2016/
[11] https://github.com/tangkk/chordscale

chords or chord progressions. They can be well captured by sequential models such as the RNN using the chordal context as the starting cue, with the discriminative objective being whether the sequence of notes are artistically "acceptable" or not. This is the discriminative part of the model, and it is similar to the LSTM semantic analysis model by Maas et al.[MDP+11].

Note that the discriminative model cannot capture generative details. Hence there should be a generative model inserted, and thus it becomes a discriminative-generative model. Referring to Boulanger's RNN-RBM based symbolic music generation system [BLBV12], if an instance of DBN (instead of RBM) is inserted between the output layer and the LSTM layer of the previous discriminative model, the resulting network will be able to both classify a sequence to labels and generate a sequence based on the prior labels (Figure 5.15).

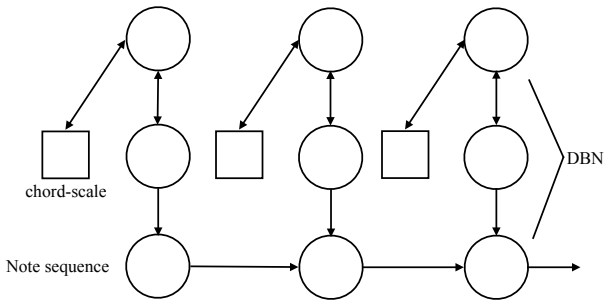

Figure 5.15 The RNN-DBN for note sequence generation

In this case, there has to be a set of training cases for every chord-scale. By training the model with jazz lick patterns, a sequence of "acceptable" notes could hopefully be automatically generated by the chord-scale sequences estimated from the backing track.

Chapter 6

Conclusion

This chapter concludes the thesis with some final remarks, and points to some possible future directions in ACE and LVACE related MIR researches. Section 6.1 will first briefly review all the contributions in this thesis; and Section 6.2 will suggest some possible future works that follow up the current state of this thesis.

6.1 Major Findings

The major research focus of this thesis is large vocabulary automatic chord estimation, or LVACE. The motivation of this research originates from the way musicians approach towards chord annotation, which is well described in Chapter 1, that:

> ...those chords are often captured in great detail, with a very large vocabulary including the suspensions, extensions, inversions and alterations.

Therefore it is necessary for an ACE system to incorporate a large vocabulary as a presumption to pass the *Turing test*, which is the ultimate goal of any kind of artificial intelligence.

However, most ACE approaches to date do not support large vocabulary. Instead, they normally use a much smaller *majmin* vocabulary due to various reasons, including the lack of training data for uncommon and long-tail chords, and the pursuing for higher evaluation scores. In order to make up for these gaps, the thesis devises some novel solutions for LVACE. The contributions of this thesis are mainly in Chapter 3 4 and 5.

Chapter 3 proposes a hybrid GMM-HMM (segmentation) and deep neural nets (classification) LVACE system framework. It assigns the segmentation and classification as two different processes. There are three deep neural nets under consideration: FCNN, DBN and BLSTM-RNN. The key finding from the experiment results is that the BLSTM-RNN implementation has significantly better performance in terms of the *ACQA* than all other considered implementations. Therefore, this model has great potential for a fully chord-performance balanced LVACE system.

Chapter 4 proposes a BLSTM-RNN with an even chance training scheme as another LVACE solution. The BLSTM-RNN performs both segmentation and classification over a piece of notegram or chromagram. The even chance training scheme gives the network much more attention to the uncommon and long-tail chords. Results demonstrate that using the scheme has significantly positive effects on the *ACQA*, and the best implementation of this approach significantly outperforms the baseline system.

Chapter 5 applies the GMM-HMM-DNN LVACE approach to jazz. Surprisingly, the preliminary experiment finds that the classical GMM-HMM segmentation still works very well in jazz. Together with the DNN classification, the best proposed jazz ACE system annotates chords much better than the

baseline approach. Note that this experiment is conducted using a test set of 7 pure jazz backings.

6.2 Future Directions

Extrapolating from this thesis, the future directions of ACE, in a deep learning perspective, should focus on at least the following three challenges:

- implementing deeper models
- embracing ground truth subjectivity
- improving accuracy on uncommon and long-tail chords

Deeper Models — The first aspect considers using a deeper model for sequence modeling. This is rather a challenge motivated by pure scientific interest than practical interest. This thesis describes a BLSTM-RNN that models sequence transformation from chromagram to segmented chord sequence. But as we know, the promise of deep learning is to extract useful features from raw input, and to learn better transformations than handcrafted transformations. Thus this challenge asks for a deep learning model in a truly "end-to-end" sense, that captures every transformation from waveform level all the way to segmented chord sequence level. Such a model might not outperform the state-of-the-art at the beginning, but might show promising potentials to achieve the optimal performance in the long run.

Subjectivity Issue — The second challenges are fundamental to all kinds of machine learning researches, and it is especially non-neglectable in LVACE

research, since human chord annotators may disagree a lot, especially in uncommon and long-tail chords. As a result, some researchers doubt the necessity to estimate uncommon and long-tail chords. Such opinions may have their merit under certain circumstances, particularly when they are referring to an ACE system built for chord learning beginners. However, they are not justified under a more practical scenario, where people need more "accurate" chord annotations for busking, practicing or rehearsal.

Therefore the annotation subjectivity issue needs to be embraced, or resolved, rather than neglected or abandoned. This is a difficult topic that may demand a lot of work and intelligence from data science, statistics, and machine learning. One key observation is that human learns to annotate chords by reading and listening to a lot of different examples, regardless of the annotators. When a person reaches a certain level, s/he may be able to correct the wrong annotations. The more advanced s/he is, the more uncommon and long-tail chords s/he is able to annotate, spot and correct.

But this issue should not overrule another equally important issue: to build a machine that "learns well". A 100% "golden standard" ground truth should not be a necessary premise for building any machine learning system. Should there be any chaos or noise, the research community could always regress to use annotations from one single annotator.

Uncommon and Long-tail Chords Accuracy — The uncommon chords accuracy is always interleaved with subjectivity issue. But it should be noted that not all instances of all uncommon chord types are subjected to the annotation subjectivity. Although currently there is not any statistics showing the exact relationship between them, musical experience told us that some uncommon

chord instances are much easier to be recognized than others, thus they receive less disagreement among annotators.

6.3 LVACE For All

In Chapter 1 we mentioned that the research scope of this thesis is mainly the LVACE for pop and rock music. In Chapter 5 we see that the same approach can actually also be applied to jazz. This leads to a believe that the same approach could also be applied to some other genres, such as hip-hop, Latin and classical music.

The first and foremost challenge of adopting the existing LVACE approach to these genres is of course the collection of ground-truth training data. But beyond that, we believe there remains a lot of challenges in these genres. For example, in hip-hop, the "vocal" is actually the "rap", which is basically speech signal. Thus the same feature extraction techniques that target pitch signals may not apply anymore. In classical music, chords are not always segmented clearly and regularly, and the vocabulary could be even much larger than those in jazz. Thus on one hand we may not be able to use the old segmentation techniques, and on the other hand we may need a new classification method to handle a lot more skewed distributed classes.

Some may propose that we build a system to segment and recognize "all chords" in any piece of music, regardless of the genres. This is still a very difficult task, if not impossible. From a machine learning perspective, we do not currently have enough data for all chords, particularly, the extension chords (such as 9, $maj13$, $min11$, etc.), inversions and alterations (such as 7b5, $m7b5$, 9#11, etc.), not to mention those from the quartal harmony, quintal

harmony or even more rare (in a common sense) chords (by data, we mean the different renditions of each of these chords by different people, instruments and groups under different musical contexts). Therefore it is impossible to build a machine learning based system for this purpose using the current data available. On the other hand, from a domain knowledge perspective, it is also very difficult. For example, one of the main challenges for an expert system in this "all chords" scenario is that it is hard to differentiate a true chord tone from a passing tone, or a tone that belongs to the previous chord or the next chord. As long as the chord segmentation remains non-perfect, expert systems will find very difficult in this "all chords" challenge.

It is truly amazing how a music virtuoso could differentiate all chords in all types of music. LVACE systems still have a long way to go before they can be compared with a music virtuoso.

List of References

[AFPZ13] Jakob Abeßer, Klaus Frieler, Martin Pfleiderer, and Wolf-Georg Zaddach. Introducing the jazzomat project-jazz solo analysis using music information retrieval methods. In *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research (CMMR) Sound, Music and Motion, Marseille, France*, pages 187–192, 2013.

[ANS+05] Samer Abdallah, Katy Noland, Mark Sandler, Michael A Casey, Christophe Rhodes, et al. Theory and evaluation of a Bayesian music structure extractor. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, ISMIR*, 2005.

[Bar04] James Murray Barbour. *Tuning and temperament: A historical survey*. Courier Corporation, 2004.

[BdHP14] J Ashley Burgoyne, W Bas de Haas, and Johan Pauwels. On comparative statistics for labelling tasks: What can we learn from MIREX ACE 2013. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 525–530, 2014.

[Bel07] Juan Pablo Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, ISMIR 2007*, volume 7, pages 239–244, 2007.

[Ben09] Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[BF03a] Tina Blaine and Sidney Fels. Collaborative Musical Experiences for Novices. *Journal of New Music Research*, 32(4):411–428, 2003.

[BF03b] Tina Blaine and Sidney S. Fels. Contexts of Collaborative Musical Experiences. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 129–134, 2003.

[BLBV12] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[BLBV13] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR*, pages 335–340, 2013.

[BMS+00] Juan Pablo Bello, Giuliano Monti, Mark B Sandler, et al. Techniques for automatic music transcription. In *Proceedings of the 1st International Society for Music Information Retrieval Conference, ISMIR*, 2000.

[Bor42] Edwin Garrigues Boring. *Sensation and perception in the history of experimental psychology*. Appleton-Century-Crofts New York, 1942.

[BP05] Juan Pablo Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, ISMIR*, volume 5, pages 304–311, 2005.

[BPKF07] John Ashley Burgoyne, Laurent Pugin, Corey Kereliuk, and Ichiro Fujinaga. A cross-validated study of modelling strategies for automatic chord recognition in audio. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, ISMIR*, pages 251–254, 2007.

[BPSW70] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

[Bro91] Judith C Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[BS05] John Ashley Burgoyne and Lawrence K Saul. Learning harmonic relationships in digital audio with Dirichlet-based hidden Markov models. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, ISMIR*, pages 438–443, 2005.

[Bur] Gary Burton. Gary Burton Jazz Improvisation Course. `https://www.coursera.org/learn/jazz-improvisation/`. Accessed: 2016-02-16.

[BWF11] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, volume 11, pages 633–638, 2011.

[CB09] Taemin Cho and Juan P Bello. Real-time implementation of HMM-based chord estimation in musical audio. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 16–21. Citeseer, 2009.

[CB11] Taemin Cho and Juan P Bello. A feature smoothing method for chord recognition using recurrence plots. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, 2011.

[Che00] Elaine Chew. *Towards a mathematical model of tonality*. PhD thesis, Massachusetts Institute of Technology, 2000.

[Cho14] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.

[CJK04] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[CLS10] Chris Cannam, Christian Landone, and Mark Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1467–1468. ACM, 2010.

[CMD+13] Chris Cannam, Matthias Mauch, Matthew EP Davies, Simon Dixon, Christian Landone, Katy Noland, Mark Levy, Massimiliano Zanoni, Dan Stowell, and Luis A Figueira. MIREX 2013 entry: Vamp plugins from the centre for digital music, 2013.

[CML07] Benoit Catteau, Jean-Pierre Martens, and Marc Leman. A probabilistic framework for audio-based tonal key and chord recognition. In *Advances in data analysis*, pages 637–644. Springer, 2007.

[Coo01] Perry R. Cook. Principles for Designing Computer Music Controllers. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 3–6, 2001.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CWB10] Taemin Cho, Ron J Weiss, and Juan Pablo Bello. Exploring common variations in state of the art chord recognition systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–8, 2010.

[CYL+08] Heng-Tze Cheng, Yi-Hsuan Yang, Yu-Ching Lin, I-Bin Liao, and Homer H Chen. Automatic chord recognition for music classification and retrieval. In *IEEE International Conference on Multimedia and Expo*, pages 1505–1508. IEEE, 2008.

[DHS12] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[DK16] Junqi Deng and Yu-Kwong Kwok. MIREX 2016 submission: Large vocabulary automatic chord estimation with deep neural networks, 2016.

[Dow08] J Stephen Downie. The music information retrieval evaluation exchange (2005-2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[DS07] Karin Dressler and Sebastian Streich. Tuning frequency estimation using circular statistics. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, ISMIR 2007*, pages 357–360, 2007.

[DY14] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[EC07] Daniel PW Ellis and C Cotton. The 2007 labrosa cover song detection system. *Music Information Retrieval Evaluation Exchange (MIREX)*, 2007.

[Elm90] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[Few10] Garrison Fewell. *Jazz Improvisation For Guitar - A Harmonic Approach*. Berklee Press, 2010.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[Fri37] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

[fS75] International Organization for Standardization. *ISO 16:1975 Acoustics âĂŞ Standard tuning frequency (Standard musical pitch)*. International Organization for Standardization, 1975.

[Fuj99] Takuya Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the 25th International Computer Music Conference*, volume 1999, pages 464–467, 1999.

[G+85] Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. Springer Berlin, 1985.

[GBD92] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[GM99] Masataka Goto and Yoichi Muraoka. Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, 27(3):311–335, 1999.

[Góm06a] Emilia Gómez. Tonal description of music audio signals. *Department of Information and Communication Technologies*, 2006.

[Góm06b] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *Journal on Computing*, 18(3):294–304, 2006.

[Gra12] Alex Graves. *Supervised sequence labelling*. Springer, 2012.

[HAHFBR01] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice Hall PTR, 2001.

[HAJ90] Xuedong D Huang, Yasuo Ariki, and Mervyn A Jack. *Hidden Markov models for speech recognition*. Edinburgh university press Edinburgh, 1990.

[Har10] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London, 2010.

[HB12] Eric J Humphrey and Juan P Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 357–362. IEEE, 2012.

[HB15] Eric J Humphrey and Juan P Bello. Four timely insights on automatic chord estimation. In *Proceedings of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2015.

[HBL13] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[HCB12] Eric J Humphrey, Taemin Cho, and Juan P Bello. Learning a robust tonnetz-space transform for automatic chord recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 453–456. IEEE, 2012.

[HDFN95] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

[HDZ+15] Xiping Hu, Junqi Deng, Jidi Zhao, Wenyan Hu, Edith C-H Ngai, Renfei Wang, Johnny Shen, Min Liang, Xitong Li, Victor Leung, et al. SAfeDJ: A crowd-cloud codesign approach to situation-aware music delivery for drivers. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(1s):21, 2015.

[HE09] Hermann LF Helmholtz and Alexander J Ellis. *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. Cambridge University Press, 2009.

[HE10] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR*, pages 339–344. Utrecht, The Netherlands, 2010.

[Hin10] Geoffrey Hinton. A practical guide to training restricted Boltzmann machines. *Momentum*, 9(1):926, 2010.

[HKO04] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[HM13] Tom Hojnacki and Joe Mulholland. *The Berklee Book of Jazz Harmony*. Berklee Press, 2013.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS05] Christopher Harte and Mark Sandler. Automatic chord identifcation using a quantised chromagram. In *Audio Engineering Society Convention 118*. Audio Engineering Society, 2005.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[HSAG05] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, ISMIR*, volume 5, pages 66–71, 2005.

[HSG06] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26. ACM, 2006.

[HSM+00] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[Hum15] Eric Humphrey. *An Exploration of Deep Learning in Content-based Music Informatics.* PhD thesis, New York University, 2015.

[IFP16] IFPI. *Global Music Report 2016.* 2016.

[Jar95] Topi Jarvinen. Tonal Hierarchies in Jazz Improvisation. *Music Perception*, pages 415–437, 1995.

[JGKM11] Nanzhu Jiang, Peter Grosche, Verena Konz, and Meinard Müller. Analyzing chroma feature types for automated chord recognition. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio.* Audio Engineering Society, 2011.

[Jol02] Ian Jolliffe. *Principal component analysis.* Wiley Online Library, 2002.

[Jor86] Michael I Jordan. Attractor dynamics and parallellism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Meeting of the Cognitive Science Society*, pages 531–546. Lawrence Erlbaum Associates, 1986.

[KdHV15] Hendrik Vincent Koops, W Bas de Haas, and Anja Volk. Integration of crowd-sourced chord sequences using data fusion. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, 2015.

[Kha11] Maksim Khadkevich. *Music signal processing for automatic extraction of harmonic and rhythmic information.* PhD thesis, University of Trento, 2011.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[Kla06] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the 7th International Society for Music Information Retrieval Conference, ISMIR 2006*, pages 216–221, 2006.

[Kle04] Lawrence A Klein. *Sensor and data fusion: a tool for information assessment and decision making*, volume 324. Spie Press Bellingham, 2004.

[KO09] Maksim Khadkevich and Maurizio Omologo. Use of hidden Markov models and factored language models for automatic chord recognition. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 561–566, 2009.

[KO11] Maksim Khadkevich and Maurizio Omologo. Time-frequency reassigned features for automatic chord recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 181–184. IEEE, 2011.

[KON] KONTAKTS. `http://www.native-instruments.com/en/products/komplete/synths-samplers/kontakt-5/`. Accessed: 2014-1-14.

[KW11] Nick Kruge and Ge Wang. MadPad : A crowdsourcing system for audiovisual sampling. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 185–190, 2011.

[KW16a] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*, 2016.

[KW16b] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. In *In Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, 2016.

[LB95] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[LBOM12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[Lee06a] Kyogu Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *Proceedings of the International Computer Music Conference*, page 26, 2006.

[Lee06b] Kyogu Lee. Identifying cover songs from audio using harmonic representation. *extended abstract, Music Information Retrieval eXchange task, Victoria, BC, Canada*, 2006.

[Lee07] Kyogu Lee. A system for automatic chord transcription from audio using genre-specific hidden Markov models. In *International Workshop on Adaptive Multimedia Retrieval*, pages 134–146. Springer, 2007.

[Lev11] Mark Levine. *The jazz theory book*. " O'Reilly Media, Inc.", 2011.

[LH95] Charles L Lawson and Richard J Hanson. *Solving least squares problems*, volume 15. SIAM, 1995.

[LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

[LS08] Kyogu Lee and Malcolm Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):291–301, 2008.

[LWH90] Kevin J Lang, Alex H Waibel, and Geoffrey E Hinton. A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1):23–43, 1990.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[Mau10] Matthias Mauch. *Automatic chord transcription from audio using computational models of musical context*. PhD thesis, School of Electronic Engineering and Computer Science Queen Mary, University of London, 2010.

[McV13] Matthew McVicar. *A Machine Learning Approach to Automatic Chord Extraction*. PhD thesis, University of Bristol, 2013.

[MD08] Matthias Mauch and Simon Dixon. A discrete mixture model for chord labelling. In *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR*, pages 45–50, 2008.

[MD10a] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR*, pages 135–140, 2010.

[MD10b] Matthias Mauch and Simon Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289, 2010.

[MDH+07] Matthias Mauch, Simon Dixon, Christopher Harte, et al. Discovering chord idioms through beatles and real book songs. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, ISMIR*, 2007.

[MDP+11] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[MFG10] Matthias Mauch, Hiromasa Fujihara, and Masataka Goto. Lyrics-to-audio alignment and phrase-level segmentation using incomplete internet-style chord annotations. In *Proceedings of the 7th Sound and Music Computing Conference (SMC)*, volume 27, pages 28–38, 2010.

[MND09] Matthias Mauch, Katy Noland, and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR*, pages 231–236, 2009.

[MSRNDB14] Matt McVicar, Raúl Santos-Rodríguez, Yizhao Ni, and Tijl De Bie. Automatic chord estimation from audio: A review of the state of the art. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(2):556–575, 2014.

[Mur02] Kevin Patrick Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[Mus04] Various Jazz Musicians. *The Real Book: Sixth Edition*. Hal Leonard Corporation, 2004.

[MXKS04] Namunu C Maddage, Changsheng Xu, Mohan S Kankanhalli, and Xi Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 112–119. ACM, 2004.

[NAW01] S Hamid Nawab, Salma Abu Ayyash, and Robert Wotiz. Identification of musical chords using constant-Q spectra. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 3373–3376. IEEE, 2001.

[NG07] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*, pages 681–688. ACM, 2007.

[Ng11] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[NMSRDB12] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[NMSRDB13] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. Understanding effects of subjectivity in measuring chord estimation accuracy. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(12):2607–2615, 2013.

[NS09] Katy Noland and Mark Sandler. Influences of signal processing, tone profiles, and chord progressions on a model for estimating the musical key from audio. *Computer Music Journal*, 33(1):42–56, 2009.

[NU87] B Nettles and A Ulanowsky. *Harmony 1–4*, pages 33–35. Boston, Massachusetts: Berklee College of Music, 1987.

[OGF09] Laurent Oudre, Yves Grenier, and Cédric Févotte. Template-based chord recognition: Influence of the chord types. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 153–158, 2009.

[OMLR+08] Nobutaka Ono, Kenichi Miyamoto, Jonathan Le Roux, Hirokazu Kameoka, and Shigeki Sagayama. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. In *Signal Processing Conference, 2008 16th European*, pages 1–4. IEEE, 2008.

[Ope] OpenCV. `http://opencv.org/`.

[Oud10] Laurent Oudre. *Template-based chord recognition from audio signals*. PhD thesis, TELECOM ParisTech, 2010.

[OWN83] Alan V Oppenheim, Alan S Willsky, and Syed Hamid Nawab. *Signals and systems*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1983.

[Pap10] Hélene Papadopoulos. *Joint estimation of musical content information from an audio signal*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2010.

[Pea14] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[PM10] Johan Pauwels and Jean-Pierre Martens. Integrating musicological knowledge into a probabilistic framework for chord and key extraction. In *Audio Engineering Society Convention 128*. Audio Engineering Society, 2010.

[PM14] Johan Pauwels and Jean-Pierre Martens. Combining musicological knowledge about chords and keys in a simultaneous chord and local key estimation system. *Journal of New Music Research*, 43(3):318–330, 2014.

[PP07] Hélene Papadopoulos and Geoffroy 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.

[PP08] Hélene Papadopoulos and Geoffroy Peeters. Simultaneous estimation of chord progression and downbeats from an audio file. In *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP*, pages 121–124. IEEE, 2008.

[PP13] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 749–753. IEEE, 2013.

[Pre98a] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[Pre98b] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[PSRI09] Carlos Pérez-Sancho, David Rizo, and José M Inesta. Genre classification using chords and stochastic language models. *Connection science*, 21(2-3):145–159, 2009.

[PT12] Hélène Papadopoulos and George Tzanetakis. Modeling chord and key structure with Markov logic. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR*, pages 127–132. Citeseer, 2012.

[Rab89] Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Ran99] Don Michael Randel. *The Harvard concise dictionary of music and musicians*. Harvard University Press, 1999.

[RD06] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[RHW88] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[RK08] Matti P Ryynänen and Anssi P Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[RMG+88] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[RRH+10] Thomas Rocher, Matthias Robine, Pierre Hanna, Laurent Oudre, Y Grenier, and C Févotte. Concurrent estimation of chords and keys from audio. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, pages 141–146, 2010.

[RSN08] Johannes Reinhard, Sebastian Stober, and Andreas Nurnberger. Enhancing chord classification through neighbourhood histograms. In *2008 International Workshop on Content-Based Multimedia Indexing*, pages 33–40. IEEE, 2008.

[RUS+09] Jeremy Reed, Yushi Ueda, Sabato Marco Siniscalchi, Yuuki Uchiyama, Shigeki Sagayama, and Chin-Hui Lee. Minimum classification error training to improve isolated chord recognition. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR*, pages 609–614. Citeseer, 2009.

[SBLD15] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[Sch89] Arnold Schonberg. *Structural Functions of Harmony*. Faber & Faber, 1989.

[Sch03] Jeff Schroedl. *Hal Leonard Guitar Method - Jazz Guitar: Hal Leonard Guitar Method Stylistic Supplement Bk/online audio*. Hal Leonard, 2003.

[SD14] Siddharth Sigtia and Simon Dixon. Improved music feature learning with deep neural networks. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6959–6963. IEEE, 2014.

[SE03] Alexander Sheh and Daniel PW Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the 4th International Society for Music Information Retrieval Conference, ISMIR*, pages 185–191. International Symposium on Music Information Retrieval, 2003.

[SGH10] Joan Serra, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: background, approaches, evaluation, and beyond. In *Advances in Music Information Retrieval*, pages 307–332. Springer, 2010.

[She64] Roger N Shepard. Circularity in judgments of relative pitch. *The Journal of the Acoustical Society of America*, 36(12):2346–2353, 1964.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SIY+08] Kouhei Sumi, Katsutoshi Itoyama, Kazuyoshi Yoshii, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation. In *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR*, pages 39–44, 2008.

[SJ01] Borching Su and Shyh-Kang Jeng. Multi-timbre chord classification using wavelet transform and self-organized map neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Citeseer, 2001.

[Smo86] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

[SMR07] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723, 2007.

[SVB09] Ricardo Scholz, Emmanuel Vincent, and Frédéric Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 53–56. IEEE, 2009.

[SW05] Arun Shenoy and Ye Wang. Key, chord, and rhythm tracking of popular music recordings. *Computer Music Journal*, 29(3):75–86, 2005.

[Tem04] David Temperley. *The cognition of basic musical structures*. MIT press, 2004.

[Tie08] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484, 2005.

[Tuk49] John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[Tur50] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[Uch] Mitsuko Uchida. Mitsuko Uchida on Schoenberg's Piano Concerto. `https://www.youtube.com/watch?v=PmWRttCo7lo`. Accessed: 2014-1-3.

[UUN+10] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based approach for automatic chord detection using refined acoustic features. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[VPM08] Matthias Varewyck, Johan Pauwels, and Jean-Pierre Martens. A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 667–670. ACM, 2008.

[Wak99] Gregory H Wakefield. Mathematical representation of joint time-chroma distributions. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 637–645. International Society for Optics and Photonics, 1999.

[WD08] Jan Weil and Jean-Louis Durrieu. An HMM-based audio chord detection system: Attenuating the main melody. *The Music Information Retrieval Exchange*, 2008.

[Wei05] Gil Weinberg. Interconnected Musical Networks: Toward a Theoretical Framework. *Computer Music Journal*, 29(2):23–39, 2005.

[WEJ09] Adrian Weller, Daniel Ellis, and Tony Jebara. Structured prediction models for chord transcription of music audio. In *International Conference on Machine Learning and Applications, ICMLA*, pages 590–595. IEEE, 2009.

[Wer90] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[Whi12] Mark White. *The Practical Jazz Guitarist*. Berklee Press, 2012.

[WSDR09] Jan Weil, Thomas Sikora, Jean-Louis Durrieu, and Gaël Richard. Automatic generation of lead sheets from polyphonic music signals. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR*, pages 603–608, 2009.

[XLDJ11] Anna Xambó, Robin Laney, Chris Dobbyn, and Sergi Jordà. Multi-touch Interaction Principles for Collaborative Real-time Music Activities: Towards a Pattern Language. In *Proceedings of the International Computer Music Conference*, 2011.

[YG11] Kazuyoshi Yoshii and Masataka Goto. A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, pages 645–650, 2011.

[YKK+04] Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *Proceedings of the 5th International Society for Music Information Retrieval Conference, ISMIR*, pages 100–105, 2004.

[Zei12] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[ZG08] Xinglin Zhang and David Gerhard. Chord recognition using instrument voicing constraints. In *Proceedings of the 9th International Society for Music Information Retrieval Conference, ISMIR 2008*, pages 33–38, 2008.

[ZL15] Xinquan Zhou and Alexander Lerch. Chored detection using deep learning. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, volume 53, 2015.

[ZR07] Veronika Zenz and Andreas Rauber. Automatic chord detection incorporating beat and key detection. In *IEEE International Conference on Signal Processing and Communications, ICSPC*, pages 1175–1178. IEEE, 2007.